

AD-A056 524

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MASS
A STRUCTURAL PARADIGM FOR REPRESENTING KNOWLEDGE.(U)
MAY 78 R J BRACHMAN

F/G 5/6

UNCLASSIFIED

BBN-3605

N00014-77-C-0371

NL

1 OF 4

AD
AC 58524



Bolt Beranek and Newman Inc.

LEVEL *II*

AD A 056524

Report No. 3605

A Structural Paradigm for Representing Know

R. J. Brachman

May 1978

Submitted to:
Office of Naval Research

AD NO. *1*
DDC FILE COPY

This document has
for public release
distribution is w

78 07 10

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN 3605 3605	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A STRUCTURAL PARADIGM FOR REPRESENTING KNOWLEDGE	5. TYPE OF REPORT & PERIOD COVERED Technical Report	6. PERFORMING ORG. REPORT NUMBER No. 3605
7. AUTHOR(s) J. Brachman Ronald	8. CONTRACT OR GRANT NUMBER(s) N00014-77-C-0371 N00014-77-C-0378	9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 334P
10. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt, Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02138	11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, VA 22217	12. REPORT DATE May 78
13. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	14. NUMBER OF PAGES 315	15. SECURITY CLASS. (of this report) Unclassified
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.		17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Epistemological primitives, epistemological representations, "dattrs", Hermes, knowledge representation, message processing, nominal compounds, nominalization, representing programs, semantic networks, Structured Inheritance Networks		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report presents an associative network formalism for representing conceptual knowledge. While many similar formalisms have been developed since the introduction of the "semantic network" in 1966, they have often suffered from inconsistent interpretation of their links, lack of appropriate structure in their nodes, and general expressive inadequacy. In this paper, we take a detailed look at the history of these "semantic" nets and begin to understand their inadequacies by examining closely what their		

DDC
JUL 18 1978
INSTITUTION

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE

Unclassified

(cont'd.)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

78 07 10 070

060 100

rest
Ray
mt

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. Abstract (cont'd.)

* is presented

representational pieces have been intended to model. Based on ~~our~~ ^{this} analysis, ~~we present~~ a new type of network — the "Structured Inheritance Network" (SI-NET) — designed to circumvent common expressive shortcomings. We acknowledge "concepts" to be formal representational objects and keep "epistemological" relationships between formal objects distinct from conceptual relations between the things that the formal objects represent. The notion of an epistemologically explicit representation language is introduced to account for this distinction, and SI-Nets are offered as a particular candidate.

The Structured Inheritance formalism that we present takes a concept to be a set of functional roles tied together by a structuring gestalt. Generic concepts, describing potentially many individuals, have as their parts generic "'dattr' descriptions", which capture information about the functional role, number, criteriality, and nature of potential role fillers' and "structural conditions", which express explicit relationships between the potential role fillers, and give the functional roles their meanings. Individual concepts have explicit binding structures ("instantiated dattr's") which indicate an individual's fillers for its roles; the individual's roles are inherited from a generic concept, in terms of which it is described. Details of the representation are elaborated, including explicit role and role-filler inheritance rules. The language is then applied to two task domains: 1) the understanding of two-word nominal compounds (e.g. "computer science", "arm chair", "hockey stick"), for which we present a conceptual analysis that uses only two basic structuring techniques to explain an extensive set of compound types; we also present a new account of nominalization, based on structured inheritance; and 2) knowledge about a complex message-processing program that is implemented on several ARPA Network hosts; we attempt to account for the structure of objects in the "Hermes" program, its commands, and the interaction of the commands and objects.

In addition to detailing these uses of the structural paradigm, we review carefully its relationship to three other current representation languages -- KRL, FRL, and MDS. The surface notation, underlying data structures, and deeper epistemological import of each of these languages is examined and compared with the others.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

BBN Report No. 3605

A STRUCTURAL PARADIGM FOR REPRESENTING KNOWLEDGE

Ronald J. Brachman

May 1978

This report is a revised version of a thesis submitted to The Division of Engineering and Applied Physics of Harvard University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the subject of Applied Mathematics, May, 1977.

This research was supported in part by the Office of Naval Research under Contract No. N00014-77-C-0371 and the Defense Advanced Research Projects Agency under Contract No. N00014-77-C-0378.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Office of Naval Research or the U.S. Government.

ACCESSION for	
NTIS	White Section <input checked="" type="checkbox"/>
DDC	Buff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	SPECIAL
A	

Abstract

This report presents an associative network formalism for representing conceptual knowledge. While many similar formalisms have been developed since the introduction of the "semantic network" in 1966, they have often suffered from inconsistent interpretation of their links, lack of appropriate structure in their nodes, and general expressive inadequacy. In this paper, we take a detailed look at the history of these "semantic" nets and begin to understand their inadequacies by examining closely what their representational pieces have been intended to model. Based on our analysis, we present a new type of network -- the "Structured Inheritance Network" (SI-Net) -- designed to circumvent common expressive shortcomings. We acknowledge "concepts" to be formal representational objects and keep "epistemological" relationships between formal objects distinct from conceptual relations between the things that the formal objects represent. The notion of an epistemologically explicit representation language is introduced to account for this distinction, and SI-Nets are offered as a particular candidate.

The Structured Inheritance formalism that we present takes a concept to be a set of functional roles tied together by a structuring gestalt. Generic concepts, describing potentially many individuals, have as their parts generic "'dattr' descriptions", which capture information about the functional role, number, criteriality, and nature of potential role fillers; and "structural conditions", which express explicit relationships between the potential role fillers, and give the functional roles their meanings. Individual concepts have explicit binding structures ("instantiated dattr's") which indicate an individual's fillers for its roles; the individual's roles are inherited from a generic concept, in terms of which it is described. Details of the representation are elaborated, including explicit role and role-filler inheritance rules. The language is then applied to two task domains: 1) the understanding of two-word nominal compounds (e.g. "computer science", "arm chair", "hockey stick"), for which we present a conceptual analysis that uses only two basic structuring techniques to explain an extensive set of compound types; we also present a new account of nominalization, based on structured inheritance; and 2) knowledge about a complex message-processing program that is implemented on several ARPA Network hosts; we attempt to account for the structure of objects in the "Hermes" program, its commands, and the interaction of the commands and objects.

BBN Report No. 3605
Bolt Beranek and Newman Inc.

In addition to detailing these uses of the structural paradigm, we review carefully its relationship to three other current representation languages -- KRL, FRL, and MDS. The surface notation, underlying data structures, and deeper epistemological import of each of these languages is examined and compared with the others.

Acknowledgements

'For my part, ...people who do anything finely always inspire me to try. I don't mean that they make me believe I can do as well. But they make the thing whatever it may be, seem worthy to be done. I can bear to think my own music not good for much, but the world would be more dismal if I thought music itself not good for much. Excellence encourages one about life generally; it shows the spiritual wealth of the world.'

George Eliot
Daniel Deronda

So many people have contributed so much to this work, that I begin to feel guilty about the singular claim to authorship on the title page. To these people, who gave unsparingly of their time and energy, I would like to express my warmest appreciation.

- To Bill Woods, for hours and hours of technical and philosophical discussion, and many of the ideas to be found in these pages. It was Bill's notion of "Mnemo" that started me on this work, and has kept me excited about it for four years. The standard of excellence set by his writing kept me striving to make this document better and better.
- To John Seely Brown, for taking such a strong interest in my work, and for constantly encouraging me about its importance. John gave me the opportunity and enthusiasm to pursue the representation of Hermes in depth.
- To Tom Cheatham, for guidance and support through the last two years of this work; and to Susumu Kuno, the fourth member of my committee, for reading this document and trying to keep me honest.
- To my parents, for all of the reasons you can think of, and more; and my two brothers, on general principles. The love and respect of my family are the things that I cherish most. It is to them that this thesis is dedicated.
- To Austin Henderson, for more hours of his time than he could really afford to spend on technical discussions, layout macros, tennis, moral support. Sometimes I think that I was not my own worst critic, but that Austin was. But without his eye for technical detail and consistency, most of the unambiguous or concise thoughts to be found here would have suffered. I can't begin to thank Austin

enough for everything, especially his enthusiasm.

To Carl Brotsker, who should be held responsible for any obscure words in this text (Carl almost tempted me into using "QUIDDITY" as a network link...). For years of friendship, and endless discussions of everything from cosmology to the World Series, Carl deserves my appreciation.

To Gwen Orr, for endless faith in my ability to complete this work, hours and hours of help in its preparation, but mostly for all the things that no one else could give me...

To all of the people at BBN, for providing such a fertile and exciting research environment. Special thanks to Dick Burton for much of his time, encouragement, and of course, volleyball. And to Brian Smith (of M.I.T.) for being a fountain of energy and insight into the most important questions of knowledge representation. Thanks also to the Hermes project for the opportunity to work on a large, real program, especially to Ted Myer who first invited me to BBN.

To those others who consistently contributed to the important side of my life, and who actually made this thesis a joyful experience - especially Jack Aiello, who among other things, managed to get me into the starting lineups of several M.I.T. softball teams. Appreciation also to Stu Nunnery, for comic relief, musical entertainment, and good friendship; also to Bill Boni, Bill Weigel, and Marshall Burack. All were a constant source of much-needed distractions.

To Stone Hall '72 (Wellesley College) and the Philadelphia Flyers, for many good friends and two Stanley Cups, respectively. Both helped me keep the proper perspective during the last four years.

Finally, to all of the people who were instrumental to the preparation of this document. To Brenda Starr, Angela Beckwith, Janet Fitzpatrick, Lorraine Luca, and Beverly Tobiason for typing parts of it, to Alison Eckert, Gwen, and Austin for helping me put it together, and to Bev for constant help in trying to find Bill Woods.

Table of Contents

Abstract.iii
Acknowledgements.v
Table of Contentsvii
List of Figures.xi
 Chapter 1. Introduction.1
1.1. The semantic network.2
1.2. The issues.4
1.3. The domains7
1.4. The organization of this report8
 Chapter 2. A Look at the Evolution of Semantic Networks11
2.1. The early nets12
2.2. Case structures.21
2.3. Concern for the foundations.28
 Chapter 3. Some Methodological Points, and Two Domains.39
3.1. Assumptions and expectations40
3.2. Reflecting underlying operations -- Epistemology42
3.3. Domains.45
3.3.1. A document information consultant.47
3.3.2. Understanding Hermes54
 Chapter 4. What's in a Concept -- A New Foundation for Semantic Nets.59
4.1. SI-Net notation.59
4.1.1. Structural conditions.67
4.1.2. Inter-concept relations.75
4.1.3. Relating nominal and verbal concepts80
4.2. Some fundamental confusions.82
4.2.1. What are nodes for?.82
4.2.2. From class to concept.84
4.2.3. Property notation.85
4.2.4. The trouble with properties.87
4.2.5. Implicit import of the class membership link90
4.2.6. The need for an epistemological foundation91
4.3. Describing potential (and actual) instances.92
4.3.1. Binary relations for property description?92

4.3.2.	Describing attribute complexes -- the basic notation	.93
4.3.3.	Individuators	.99
4.3.4.	A note on the standard abbreviated notation	.101

Chapter 5. What Holds Things Together? --

	Intensions and Structure	.103
5.1.	Intensional structure	.104
5.1.1.	Intensions and concepts	.104
5.1.2.	The body of the predicate	.107
5.1.3.	Connections between dattr and structural condition	.110
5.1.3.1.	The case against case	.112
5.1.3.2.	Full roles	.114
5.1.4.	More complex structural conditions	.115
5.1.5.	More complex accesses from S/C's	.119
5.1.6.	A final note on structural conditions	.122
5.2.	Derived intensions	.123
5.2.1.	Role-oriented modification	.125
5.2.1.1.	Restriction	.125
5.2.1.2.	Role differentiation	.126
5.2.1.3.	Particularization	.127
5.2.2.	More global modifications	.129
5.2.2.1.	Modification of structural condition	.129
5.2.2.2.	Analogy	.130
5.3.	Consequences of intensional structure	.131
5.3.1.	Passing structure -- the DSUPERC link	.132
5.3.2.	Individuators as individual concepts	.133
5.3.3.	Using intensional structure to understand networks	.135

Chapter 6. Understanding Nominal Compounds137

6.1.	Understanding English nominal compounds	.139
6.2.	<u>The Grammar of English Nominalizations</u>	.145
6.2.1.	A sketch of Lees' account	.146
6.2.2.	Elements of a new analysis	.147
6.3.	Deriving nouns from verbs	.149
6.3.1.	Agents, facts, and actions	.150
6.3.2.	Nominals conceptualized	.151
6.4.	The structure of English compounds	.169
6.4.1.	Verb-plus-dattr compounds	.172
6.4.1.1.	Object-Verb compounds	.173
6.4.1.2.	V-O compounds -- Habitual activity and purpose	.177
6.4.1.3.	Subject-Verb compounds	.180
6.4.1.4.	Verb-Prepositional Object compounds	.182
6.4.2.	Noun-plus-dattr compounds	.185
6.4.3.	Dattr-dattr compounds	.189
6.5.	Understanding compounds	.193
6.5.1.	Analogies	.198

Contents

Chapter 7. Representing Knowledge about Hermes201
7.1. Internal structures for Hermes objects.202
7.2. A hierarchy for Hermes commands212
7.3. Bringing commands and objects together.220
7.4. Keeping track of the Hermes environment; Individuation. .229	
 Chapter 8. An Analysis of Current Representation Methodologies .237	
8.1. Language forms.239
8.1.1. Basic language constructs240
8.1.2. Procedures and constraints.245
8.1.3. Some special forms.249
8.2. Basic representational structures251
8.2.1. Structured conceptual objects252
8.2.2. Individuation structures.258
8.2.3. Inheritance261
8.3. Representation semantics.264
8.3.1. Meanings of basic structures.268
8.3.2. Role descriptions and structure271
8.3.4. Individuation and individutors273
8.4. Conclusions274
8.4.1. SI-Net notation as a pedagogical tool276
 Chapter 9. Conclusions279
9.1. Dattrs and structural conditions.279
9.2. The structure of a consultant's knowledge281
9.2.1. Using the representation.284
9.2.1.1. Assimilation of new information285
9.2.1.2. Paraphrase and inference.287
9.3. General use and specific shortcomings292
9.3.1. What remains to be done295
9.4. Three follow-up projects.297
 Appendix: Historical Perspective on Nominalizations299
A.1. Lees.299
A.2. Fraser's and Chomsky's nominalizations.302
 Bibliography.307

BBN Report No. 3605
Bolt Beranek and Newman Inc.

List of Figures

2.1	Quillian's "planes".13
2.2	A TLC unit16
2.3	A simple hierarchy17
2.4	SCHOLAR units.19
2.5	Structural description of an ARCH.20
2.6	A Simmons case structure22
2.7	Some Rumelhart, <u>et al.</u> concepts.23
2.8	A conceptual dependency conceptualization.24
2.9	Heidorn's "IPD".26
2.10	A HAM proposition.27
2.11	Separating system relations from item relations.30
2.12	A partitioned set of nodes31
2.13	A proposition node34
2.14	Hayes' depictions and binders.36
4.1	A simple concept63
4.2	A predicative concept.64
4.3	The function DISTANCE(x,y)65
4.4	The ARCH object.66
4.5	Incomplete structure for an ARCH68
4.6	Detailed structure of an ARCH.70
4.7	Logical operators.72
4.8	Access ambiguity73
4.9	Composite access74
4.10	An individuator and its defining concept76
4.11	Differentiated roles78
4.12	A property86
4.13	A generalized property87
4.14	Basic SI-Net notation.96
4.15	A node for COMMAND97
4.16	Roles and role chains.99
4.17	An "attribute/value" pair100
4.18	Conceptual relations.101
5.1	HYDROGEN/BOMB in terms of other concepts.109
5.2	A MESSAGE in terms of operations on it.111
5.3	Basic definition of FUEL.115
5.4	Conjunction in the structural condition117
5.5	A universal quantification.118
5.6	Skolemization120
5.7	"The name of the sender (of a message)"122
5.8	A subconcept.126
5.9	Differentiated roles.127

5.10	A particularized concept.128
5.11	Individual concepts135
6.1	A simple verb structure152
6.2	An event.152
6.3	A Factive derived from an event153
6.4	Contexts for Factives155
6.5	Properties of Factives.157
6.6	THUMB1, a green thumb158
6.7	A Factive from a role instance.159
6.8	Derived abstract nominals161
6.9	A restricted nominal.162
6.10	Factive and Activity nominals166
6.11	Derived role nominals167
6.12	Basic O-V structure174
6.13	"News broadcasting"174
6.14	Abstract and concrete COMPL-ACTION compounds.175
6.15	Object plus Agentive.177
6.16	A "purpose adverbial" -- V-O.178
6.17	S-V compounds180
6.18	A V-S compound with "purpose adverbial"182
6.19	Nominalized verbs restricted by prepositional objects184
6.20	Prepositional objects restricted by verbal modifiers.185
6.21	Nominal plus part186
6.22	Optionality in choice of modifier187
6.23	A very vague definition190
6.24	HYDROGEN POWERS a HYDROGEN/BOMB191
6.25	Derivation of CHILD/REARING194
6.26	An existing compound and a derived one.197
6.27	HOCKEY : STICK :: BASEBALL : BAT.199
7.1	A non-viable attempt at SWITCHBLOCK203
7.2	The dattrrs of SWITCHBLOCK205
7.3	SWITCHBLOCK, in detail.206
7.4	Structured substructures.207
7.5	Some MESSAGE/FIELDS209
7.6	Draft message211
7.7	Hermes' notion of COMMAND214
7.8	Transcribing commands, part 1216
7.9	Transcribing commands, part 2217
7.10	Transcribing commands, part 3219
7.11	An elementary effect.223
7.12	A first attempt at synthesis.226
7.13	Substructure reference.228
7.14	A deeper substructure reference230
7.15	An individual SWITCHBLOCK232
7.16	Some particular objects233
7.17	SURVEY RECENT235
7.18	Transformation between command aspects.236

Figures

8.1	An MDS template241
8.2	Two KRL-0 units242
8.3	KRL-1 units243
8.4	A frame for LUNCH244
8.5	A frame with an "if-added" method246
8.6	A KRL-0 call on LISP code247
8.7	An MDS consistency condition.248
8.8	Generalized "chunk" structure252
8.9	A closer look at properties in KRL.260
8.10	KRL unit/perspective structure.270
9.1	Part of the S/C of TRANSFORM/MSG/THRU/TEMPLATE.290
Table A.1	Lees, Fraser, and Chomsky on nominals.306

Chapter 1. Introduction

For many years, science fiction enthusiasts have dreamed of a future in which we are joined by intelligent machines -- the popular literature is filled with fantasies of robots, androids, and sentient mechanical servants with amazing cognitive abilities. While the earliest stories of intelligent machines were pure futuristic fancy, the arrival of the digital computer in the 1950's seemed to promise the transformation of such visions into reality.

However, despite twenty-five years of life with computers, and even the advent of a new field of study called "Artificial Intelligence", the age of robots is not yet upon us. The term "thinking machine" appears to have been a bit premature -- no "electronic brain" has yet had a thought.

While little intelligence has so far been displayed by our computers, research into automated intelligent behavior has moved slowly forward. In the late '60's and early '70's several programs appeared that seemed to possess rudimentary cognitive powers (see [Minsky 1968] for descriptions of a few) but their success was limited to tightly constrained situations and tasks that were very patterned. Recently researchers have begun to suspect that despite the seemingly awesome calculating power of the modern digital computer, it could not be expected to learn how to do things from scratch (see, for example, [Brachman 1975]). Rather than start with a blank memory, our computer must have at least some knowledge before it attempts a task requiring "intelligence" (at the least it must know how to learn). In addition, when it acquires new knowledge, the computer must store the information in some form which will enable it to be accessed at a later time. The

study of such forms has become known as the study of the representation of knowledge, and this inquiry now constitutes a critical component of research into making computers intelligent. Without a flexible, extensible, accessible representation for what it comes to "know", the electronic brain is incapacitated with amnesia.

While the recent infatuation with structures for knowledge perhaps makes it feel like we are the first to pursue the idea, this area has its roots well in the past. For centuries the limits and structure of knowledge have been studied by philosophers: what we can know and how we can come to know it have been the central questions of Epistemology. However, we do hold a new advantage: we have the computer as a laboratory in which our theories of knowledge and representation can be put to the test. By implementing a theory of knowledge on a machine, we can see if it proves an adequate model -- if it gives the computer the ability to acquire knowledge of language and the world in a way that allows it to make intelligent use of that knowledge in confronting new situations. This report is concerned with investigating the nature and limits of knowledge in computer-implementable form, a study in what might be called "epistemological engineering". In it, we will consider many of the important characteristics that a representation must exhibit to support computer programs that behave intelligently, and present a new candidate that seems to exhibit these qualities.

1.1. The semantic network

One popular computer-compatible model of human knowledge that has evolved since the middle '60's is the semantic network ([Quillian 1966, 1967, 1968, 1969] -- see Chapter 2 for other references). This graphical representation language is primarily associative. That is, one of the important features of our own memories that the original semantic net authors tried to capture is the way that our knowledge is

Section 1.1
The semantic network

highly interconnected, and forms a network of concepts, facts, and beliefs all related to one another. A good example of such interdependent information is the dictionary: each word is defined as a sequence of other words defined elsewhere in the same volume. Thus a conceptual representation would have one word's meaning pointing to a structured set of other words' meanings. In a fairly intuitive, natural way, the net structure attempted to reflect just this kind of word semantics*.

Semantic networks have been used in more and more computer implementations as representations for the knowledge of programs called upon to do increasingly sophisticated tasks. The nets are being asked to represent a wide variety of abstractions, such as "concepts", "facts", "expressions", "propositions", "meanings", etc. Unfortunately, none of the programs using these structures has convincingly demonstrated the powers of understanding that the semantic net was supposed to afford it. The thesis here is that this failure to achieve the original goal of "humanlike use" of knowledge is at least in part due to a lack of appreciation of what it is that we are attempting to represent. I believe that a clearer understanding of what "concepts" are will lead to better representations.

* A universal question here is "What's so 'semantic' about a 'semantic network'?" In my opinion, it is not so much the network that is semantic, but that the original author (Quillian [1966]; see Chapter 2) intended to capture the semantics of English words in his nets. This is an important point, because the folklore that has grown around the idea seems to be based on the belief that there is something semantic about the networks themselves; as we shall soon see, this is a misleading myth.

1.2. The issues

In this report, I will investigate in detail the shortcomings of the usual conception of a semantic net as a representation for knowledge, to try to ascertain why it is inadequate. My primary intent is to develop a new network representation that will avoid the difficulties suffered by the older nets and handle a broad range of representation phenomena that I consider to be benchmarks in the representation process.

In particular, I will here set out to resolve in a new notation the most important problem with semantic networks (an issue that I shall refer to as "foundations for semantic networks"): the foundational primitives of the most common network representation languages (i.e., "nodes" and "links") are inadequate to express what we expect "concepts" to express. This is partly due to the lack of a precise semantics for nodes and links. That is, network languages do not generally include sets of primitive node and link types which have fixed, precise interpretations*; they offer instead only the general notion of "nodes" and "links". The foundational problem is also in part due to the attempt to use a single language to represent associations between particular concepts as well as associations between the formal objects of the representation language. Network notations are usually so homogeneous as to confuse the underlying logical and epistemological operations of the formalism with the conceptual information that is to be represented in it. To resolve these foundational difficulties, I propose two things: a set of methodological suggestions for developing representations -- emphasizing detailed investigation into the objects being represented and the logical principles of the representation itself -- and an epistemology which explicitly captures the most

* This is a feature of networks that fortunately is showing some improvement. See [Schubert 1976], [Hendrix 1978], and [Levesque & Mylopoulos 1978].

fundamental elements of knowledge representation, namely, the underlying relationships between "concepts" as formal objects (i.e., part-whole relationships, structural relationships between parts, "instantiation", etc.). The epistemology that I propose will be accompanied by a set of rules that specifies how "concepts" are to be built from elementary representational units. The representation scheme based on this is an example of an "epistemologically explicit" language.

While having a representational foundation that is consistent and logically adequate is paramount, there are several other responsibilities that an adequate formalism for representing knowledge must accept. These include

- The representation of structured objects.

Most of the objects that we encounter in our everyday lives can be perceived to have internal structure (they are not "atomic" entities). Unfortunately, most semantic net representations do not attempt to handle the representation of objects in any but the most cursory way -- objects are simply taken as primitives, and are represented by nodes with no links indicating the object's internal makeup. Those representations which do represent objects with parts do so by specifying a set of "cases" for an object, relying on a fixed set of case relationships to express the internal structures that objects might have. In Chapter 5, I show how the notion of a small number of cases is inadequate to handle all of the possible relationships between the parts of objects. Advocated instead is the definition of a concept as a set of functional roles tied together with an explicit structuring interrelationship, built from other concepts existing in the network. Chapter 5 is devoted to the explication of this type of structural description, which is absent from previous knowledge representations.

- Deriving new concepts from old.

One of the fundamentals of understanding is the ability to perceive a new structure in terms of already known concepts. There are many ways in which new concepts can be defined -- for example, their parts may be similar (but not identical) to those of other concepts, or new types of parts may fit together in ways defined by known relationships. In any case, there are many types of definitional connection between concepts that must be accounted for by an adequate representation. Most network formalisms fall short in their ability to express any but the

most rudimentary inter-concept relations. In this report, I investigate in depth several different kinds of (intensional) relationships between concepts, including the important notion of individuation -- producing the description of a particular individual from a more general description of a class of individuals. I look on the proper handling of inter-concept relations as one of the keys to the formalism's capacity for assimilation of new information, since many concepts can be considered already known implicitly by virtue of their potential derivation from already existing concepts. Further, given a clear way to structure new elements from old ones, it is incumbent upon a representation to allow the inference of relationships that are implicit in the structure but not explicitly represented.

- Relating nominal and verbal concepts.

Objects and actions can be related in two important ways, both of which must be expressible in a representation: 1) a nominal concept may be derived from a verbal one, and thus may describe a process or event as an entity unto itself, and 2) an action, while in progress, may operate on an object.

- The representation of idiosyncratic interpretations.

Much of our knowledge is incomplete, vague, or stylized, and to perform intelligent activities a knowledge-based program must allow for flexibility in the definition of concepts. Each person has an idiosyncratic understanding of the concepts s/he knows; a representation must provide the means to express a concept in terms of the current set of concepts available in a particular data base (not in terms of universal "knowledge primitives").

- Paraphrase retrieval.

In many applications, requests for information from a knowledgeable program will vary somewhat from the particular way in which the desired information has been stored. Rather than force a canonical representation on the request and stored data, a knowledge-based program should be able to take advantage of definitional information in its knowledge structure, and retrieve information by paraphrasing the request.

My intent in this report is to present a new type of representation, the "Structured Inheritance Network" (SI-Net), that I expect to be able to stand up to each of these challenges. This type of representation would thus be a good candidate for the memory structure of a program

Section 1.2
The issues

that might be taught about a particular domain. Once it had assimilated the domain-specific knowledge, such a program could then serve as a consultant about the topics that were associatively connected in its memory.

1.3. The domains

In this report I will use an SI-Net formalism to begin to represent in depth two particular domains, in order to show how it can handle the complex and often subtle information that it must capture to be useful in a consulting task. The choice of the particular domain of study is in general a very important one; only realistic application tasks will expose the non-obvious weaknesses of a representation. In addition, one must push to solve the most difficult problems of the task, or s/he cannot claim to have truly represented the domain. Most often, one is not even aware of the deeper representational challenges of an application task until a great deal of effort has been expended on the hardest problems.

The two particular application tasks investigated here are fairly disparate in surface appearance, but each has important fundamental demands on a representation of knowledge. In Chapter 3, I introduce the idea of a document consultant that might ultimately read the annotations from an annotated bibliography and produce reading lists upon query by a user. There are some difficult problems in assimilating the concepts in annotations, and I address a particular one, the understanding of nominal compounds. Very often, the topics of documents are expressed as compound concepts, with no indication in the surface string of the underlying structuring relationships. I concentrate on this pivotal understanding task, first examining the idea of a "nominalization", and then outlining a scheme for representing nominal compounds.

I also introduce the idea of a program consultant, a helpful agent that might aid a user in getting to know a large computer program. The particular program that I will deal with here is "Hermes"*, a large interactive system for reading, writing, and manipulating electronic messages on the ARPA computer network. Hermes has a set of commands which operate on messages and related objects, and knowledge about its operation is complex enough to present some interesting challenges to the SI-Net paradigm.

1.4. The organization of this report

Chapter 2 follows this introduction with a semi-historical account of the development of the notion of a semantic network, so that we may appreciate the state of knowledge from which this research has emerged. The survey does not cover the entire spectrum of semantic net research, but instead tries to point out the most important aspects of the formalism's development. I cover the early nets, some projects that attempted to incorporate linguistic case structure, and finally, some important foundational studies.

Chapter 3 then tries to outline explicitly a methodological approach to investigating representations of knowledge, in line with our focus on "foundations for semantic nets". Since one of the important methodological issues is the selection of a domain to be represented, this chapter introduces the two domains in a fairly extensive discussion. Chapter 3 thus sets the stage for the two particular problems that I attempt to handle with Structured Inheritance Networks -- nominalizations and nominal compounding, and the description of a complex program. I also show how the six critical issues (Section 1.2)

* "Hermes" is a trademark of Bolt Beranek and Newman Inc. [Myer, Mooers & Stevens 1977]

arise from the needs of the two domains.

The representational paradigm that I propose is discussed in Chapters 4 and 5. First, Section 4.1 presents the SI-Net notation that I will use in its entirety; this section introduces the small set of representation pieces ("link types") and illustrates their epistemological orientation. Then, I begin a more detailed development of the representation by carefully considering the underlying operations that I wish to incorporate into it (thus following the methodology set out in Chapter 3). In Section 4.2 I analyze the capabilities of semantic networks, pointing out some important deficiencies in the standard notations. In Section 4.3 I set out to resolve the ambiguities and inadequacies exposed, and motivate the set of links for representing concepts, "dattrrs" (role/filler/context structures), and instances.

Chapter 5 discusses the "internal structure" of concepts and its implications. Here I use the philosophical notion of an intension to help better understand the relations that tie concepts together. I conclude the discussion of the SI-Net notation by illustrating how its primitives express these interrelations. Chapters 4 and 5 may be read independent of the rest of the report, as they present a self-contained discussion of the representational paradigm.

The two subsequent chapters are the representational heart of the report. In Chapter 6, I look in depth at the potential solution of a linguistic problem -- the representation of nominalizations and nominal compounds -- with the network paradigm introduced earlier. The first part of the chapter includes a discussion of the problems involved with understanding compounds and makes clear how the set of issues fits into this task. I also discuss the representation of nominalizations as a prerequisite to that of compounds, and introduce some relevant early work done by Lees [1963]. The analysis of compounds accounts for Lees' large number of syntactic categories with just two basic underlying compounding operations.

Chapter 7 illustrates how SI-Net structure can be used to represent knowledge about the Hermes computer program. Commands, objects, and their interactions are all accounted for in depth. The knowledge bases discussed in Chapters 6 and 7 cover a broad and important range of representational phenomena, and their successful treatment represents a significant step toward the eventual production of the aforementioned "consultant" programs. These two application chapters are independent of one another, and either could be read in conjunction with Section 4.1 (along with the appropriate introductory section of Chapter 3) to get a good feel for the nature of SI-Net representation.

Given the representation that has been developed in Chapters 4 and 5, and the issues that have been raised, Chapter 8 presents a detailed analysis of three current representation paradigms -- KRL [Bobrow & Winograd 1977], MDS [Irwin & Srinivasan 1975, Srinivasan 1976], and FRL [Goldstein & Roberts 1977, Roberts & Goldstein 1977]. The discussion is reasonably independent of the rest of the report, given a basic knowledge of semantic networks in general. However, Section 4.1 should be consulted, as part of the analysis is based on the SI-Net paradigm. This chapter is really a synopsis of what I feel to be the current Zeitgeist in knowledge representation, and summarizes the model presented in this report in perspective with others like it.

Finally, Chapter 9 is an attempt to summarize the contributions of this work in its own right, and includes some suggestions for future work which might grow out of the research presented here. In particular, I try to assess how far what I have reported here has gotten us towards the realization of an intelligent consultant program.

Chapter 2
A Look at the Evolution of Semantic Networks

Chapter 2. A Look at the Evolution of Semantic Networks

The idea of a memory based on the notion of associations is apparently a very old one -- Anderson and Bower [1973, p. 16] trace the idea back as far as Aristotle. However, only recently has the associative memory idea taken a firm hold with those interested in modeling human memory or providing working memories for intelligent computer programs. In this chapter, I would like to summarize several of the recent projects which have set the stage for the research described in this report.

The last ten years have seen a tremendous explosion in the number of efforts directed toward developing memory models which might be considered networks, and the literature has expanded to the point where only with extreme effort can one maintain familiarity with the entire field. To treat fairly all of the work that has led to our current state of knowledge about knowledge representation would be a Herculean task, and one requiring far more space and time than is convenient here. Therefore, my analysis will begin with Ross Quillian's [1966] work, and will not discuss the many earlier efforts of Gestalt psychology, perception-by-reconstruction theories (especially [Bartlett 1967] and [Neisser 1967]), and Artificial Intelligence that have had significant effects on the current shape of semantic nets. I will only briefly outline the various major contributions to the semantic network

This chapter appears as the first section of "On The Epistemological Status of Semantic Networks", in Associative Networks -- The Representation and Use of Knowledge in Computers, edited by Nicholas V. Findler (New York: Academic Press). It has been updated from the original chapter in the dissertation to include some important recent work.

literature, and hope that the bibliography at the end of this report will provide sufficient direction for the reader more interested in historical trends and details on the representations sketched here. I will not proceed strictly chronologically (many of these projects developed simultaneously), but will instead broadly outline three major groups of work -- the early nets that provided the basic structure, those which attempted to incorporate linguistic case structure, and several more recent important foundational studies. In addition to this more shallow survey, Chapter 8 will provide a detailed analysis of three of the most important contemporary projects, which are developing representations different in appearance, but similar in spirit to semantic nets.

2.1. The early nets

The idea of a "semantic network" representation for human knowledge is generally acknowledged to have originated in the work of Ross Quillian [1966, 1967, 1968, 1969, Bell & Quillian 1971]; Quillian proposed an associational network model of "semantic memory" in his Ph.D. thesis in 1966. His intent was to capture in a formal representation the "objective" part of the meanings of words so that "humanlike use of those meanings" would be possible [1966, p. 1]. The representation was composed of nodes, interconnected by various kinds of associative links, and closely reflected the organization of an ordinary dictionary. The nodes were to be considered "word concepts", and links from a concept node pointed to other word concepts which together made up a definition, just as dictionary definitions are constructed from sequences of words defined elsewhere in the same volume. The structure thus ultimately became an interwoven network of nodes and links.

In Quillian's structure, each word concept node was considered to be the head of a "plane" which held its definition. Figure 2.1 [Quillian

Section 2.1
The early nets

1968, p. 236] illustrates a set of three planes (indicated by solid

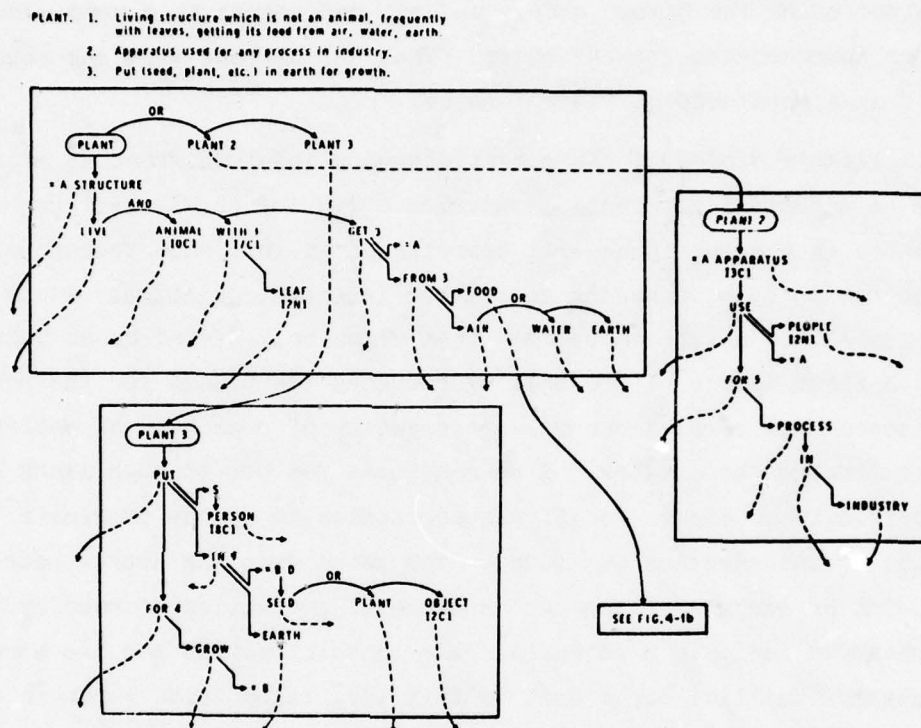


Figure 2.1. Quillian's "planes".

boxes) for three senses of the word, "plant". Pointers within the plane (the solid links in the figure) are those which form the structure of the definition; Quillian postulated a small set of these, which included subclass (e.g., the relationship of PLANT2 to APPARATUS in the figure), modification (e.g., APPARATUS is modified by the USE structure), disjunction (labelled by "OR"), conjunction (labelled by "AND"), and subject/object (e.g., the parallel links from USE to PEOPLE (the subject) and to =A (the object)). Pointers leading outside the plane (the broken links in the figure) indicate other planes in which the referenced words are themselves defined. The fact that in Quillian's structure words used in definitions of other words had their

own planes, which were pointed to by place-holder nodes within the definition, corresponded to the important "type/token" distinction. Each word was defined in only one plane in the structure (the head of the plane being the "type" node), and all references to a word went through intermediate "token" nodes. Thus definitions were not repeated each time a word concept was referenced.

Quillian's desire of his semantic memory model was that it might serve as a general inferential representation for knowledge. He presented in his thesis several examples of an inference technique based on the notion of a spreading activation intersection search -- given two words, possible relations between them might be inferred by an unguided, breadth-first search of the area surrounding the planes for the words; this search was carried out by a propagation of some kind of activation signal through the network. A search would fan out through links from the original two planes to all planes pointed to by the originals, until a point of intersection was found. The paths from the source nodes to the point of contact of the two "spheres of activation" formed by the search would indicate a potential relationship between the two word concepts*. Quillian hoped that in this way, information input in one frame of reference might be used to answer questions asked in another. The use of information implicit in the memory, but not stated explicitly, was one of the important features of the memory model.

Part of the reason that certain properties could be inferred from such a memory was its use of a link indicating a "subclass" relationship and a link specifying a "modifies" relation. A concept could be defined in terms of a more general concept (of which it was a subclass) and a modifying property, which was a combination of an attribute and a

* The belief that properties of a node could be found by an expanding search led Quillian to the idea that a word concept's "full meaning" comprised everything that could be reached from the patriarchal type node (the head of its defining plane) by an exhaustive tracing process.

particular value for that attribute*. In this characterization, properties true of a class were assumed true of all of its subclasses, except for the modifications. As a result, the superclass chain extending upward from a concept embodied all of the properties true of that concept. Thus the semantic net represented the combination of two important types of memory feature -- a superclass-subclass taxonomic hierarchy, and the description of properties (attribute/value pairs) for each class. Earlier work done by Lindsay (see [Lindsay 1973] for a later discussion of Lindsay's original work) and Raphael [1968] can be seen to be the precursors of this important marriage.

Quillian later cleaned up his memory model a bit. He eliminated the type/token distinction by making everything in the net a pointer, and, in a project called the "Teachable Language Comprehender" (TLC) [1969], he investigated its utility as a knowledge base for the reading of text. In TLC, a property was formally defined to be an attribute (some relational concept), a value, and possibly some further "subproperties". Properties were used in the definitions of "units", which represented the concepts of objects, events, ideas, assertions, etc.: a unit was defined by its superset and a set of refining properties. For reading, an intersection technique was used to find relations between words encountered in a text (this was augmented by the application of certain "form tests" as syntax checks). Figure 2.2 [Quillian 1969, p. 462] illustrates a simple unit. The unit being defined in this figure is the one for "client". The unit indicates that a CLIENT is a PERSON (i.e., PERSON is its superset), with a further qualification indicated by the second pointer from the unit to a restricting property. That property combines the "attribute", EMPLOY, with a value, PROFESSIONAL, and the subproperty, "BY the CLIENT".

* Quillian claimed that his nodes corresponded "to what we ordinarily call 'properties'" [1966, p. 26].

BBN Report No. 3605
Bolt Beranek and Newman Inc.

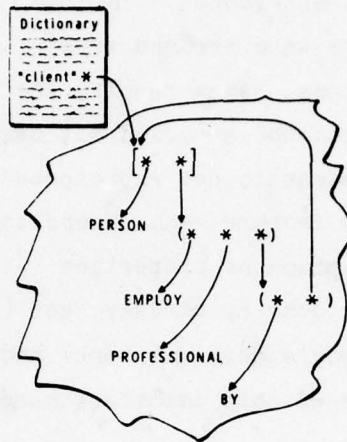


Figure 2.2. A TLC unit.

While TLC was an interesting model for finding connections between word meanings, its success in reading was limited. TLC's failure to achieve understanding was at least in part due to its insufficient set of link types and the fact that the search did not take into account the meanings of the various links. Despite the many shortcomings of his model, however, Quillian's early papers contain the seeds of most of the important ideas that are today the mainstays of semantic nets.

Quillian's revised TLC format gave rise to two other important studies. With Allan Collins, Quillian himself undertook a series of experiments to test the psychological plausibility of his network scheme [Collins & Quillian 1969, 1970a, 1970b, 1972a*], and the networks they used to check reaction time are easily recognized as the direct forerunners of recent networks (see Fig. 2.3 [Collins & Quillian 1970a, p. 305]). The nets were simple superset hierarchies of concepts like "Animal", "Bird", and "Canary", with each node having attached a set of

* The reader is also referred to an interesting article by Collins and Quillian called "How to make a language user" [1972b], in which they summarize many of the things that they learned from their experiments about language and memory.

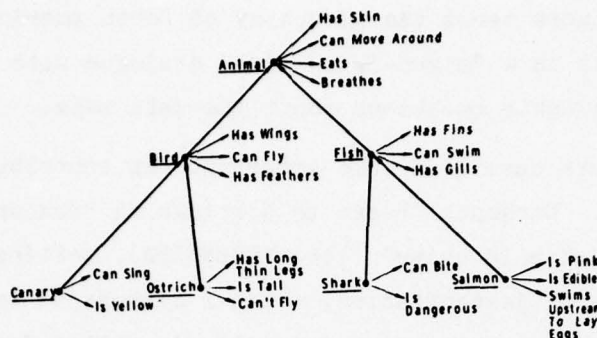


Figure 2.3. A simple hierarchy.

properties defining its corresponding concept (e.g., "has skin", "has wings", "is yellow", etc.). Since more general properties were supposedly stored higher up the generalization hierarchy, one would expect it to take more time to affirm a statement like "A canary has skin" than one like "A canary is yellow." The reaction time studies seemed to confirm the plausibility of such a hierarchical model for human memory, although not conclusively. In any case, the experiments crystallized the notion of inheritance of properties in a semantic net (the passing of values like "has skin" from the general concept "Animal" to the more specific "Canary"), and gave rise to a concrete notion of semantic distance between concepts (i.e., the number of links to be traversed between two nodes). More recently, Collins and Loftus [1975] have discussed in much detail the psychological implications of an extended version of this model, and have examined some experimental results in regard to their "spreading-activation" theory of processing (a sophistication of Quillian's semantic intersection technique). The reader is referred to that paper for some clarification of Quillian's original theory and a defense of the original experiments.

The other significant project arising directly from Quillian's TLC work was established by Carbonell [1970a, 1970b] and attempted to use Quillian's networks as a data structure in an implemented computer-aided instruction program. The SCHOLAR program had a knowledge base which

described in network terms the geography of South America. A student could participate in a "mixed-initiative" dialogue with the system, asking and being asked questions about the data base.

SCHOLAR's data base made some important new contributions to Quillian's nets. Carbonell began to distinguish "concept units" (like LATITUDE) from "example units" (like ARGENTINA), setting the stage for the later notion of instantiation, which I discuss at length in this report*. In addition, a notion of Quillian's called "tags" was expanded and used extensively. Figure 2.4 [Carbonell 1970b, p. 194] illustrates the SCHOLAR units for latitude and Argentina; in the text part of the figure, the name of a unit follows "RPAQQ" (a LISP value-setting function), and anything within the unit that follows a left parenthesis is an attribute. Tags on relations are indicated by parenthesized pairs following the attribute names (e.g., the "SUPERP" of LATITUDE is LOCATION, and has the tag "(I 2)"). The most important of the tags in SCHOLAR was the "irrelevancy tag" ("I-tag"), which could explicitly increase the semantic distance between two nodes. I-tags were used to determine the relevance of certain facts in a given context, and allowed the system to start with the most relevant aspects of a unit when describing a concept to the student. In addition, SCHOLAR introduced temporary, time-dependent tags. Also, while SCHOLAR's units looked much like Quillian's TLC units, the properties associated with a unit had as

* Instantiation has become one of the most well-known aspects of semantic net formalisms. The general idea is the association of a particular individual with the class of which it is a member, and in most notations, this is reflected by the construction of an individual description based on a generic description that the individual satisfies. Thus, while we primarily think of instances as things in the world which are manifestations of our abstract concepts, the term "instantiation" is very often used to refer to the production of a description of an individual based on a more general description. I will later use the term "individuation" (of description) for this latter intent, avoiding the potential confusion over what the term "instance" really means. However, in this chapter I will continue to use "instantiation", since it is the term used by all of these authors.

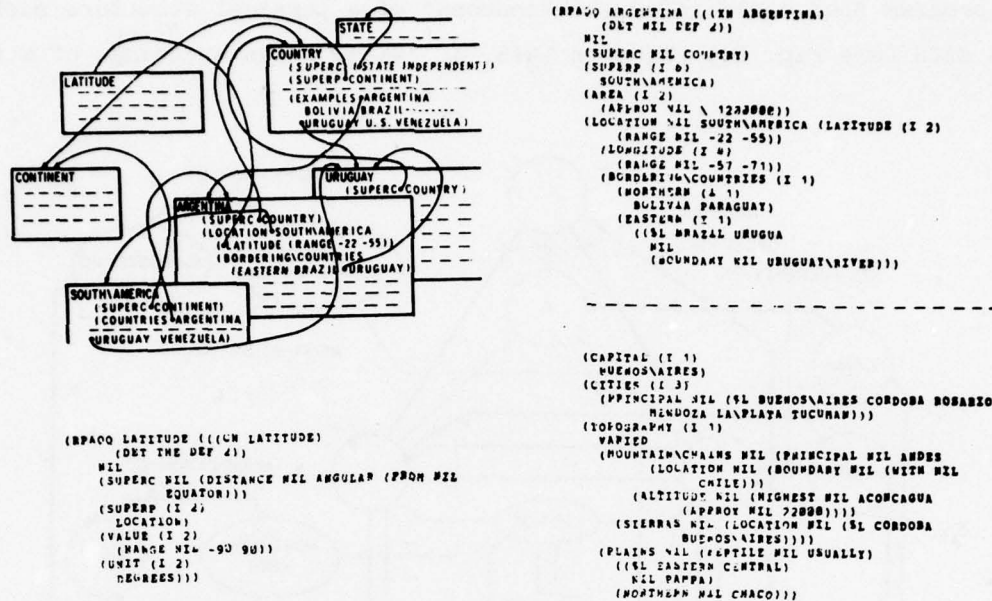


Figure 2.4. SCHOLAR units.

their first elements the names of attributes, rather than pointers (resurrecting the type/token distinction). Thus the precedent was set for naming links -- associating arbitrary labels with the associations between units. In addition to several special attributes ("SUPERP" for superconcept, "SUPERP" for superpart, and "SUPERA" for superattribute), things like "LOCATION", "TOPOGRAPHY", "CITIES", "UNIT", etc. were now being encoded directly into the network*. Another important precedent set in the SCHOLAR net was the intermixing of procedures with the declarative structure. LISP functions associated with units were used to actively infer properties that were not stated as declarative facts.

Another early effort, which proceeded independently of the Quillian/SCHOLAR work but made use of similar structures, was Winston's "structural descriptions" work at M.I.T. [1970, 1975]. Winston created

* While Carbonell claimed that no links were privileged [1970a, p. 112], I shall show later how those like "superp" are very special indeed.

a program that could infer the "concept" of a physical structure such as an ARCH (see Fig. 2.5 [Winston 1975, p. 198]), given encodings of a set

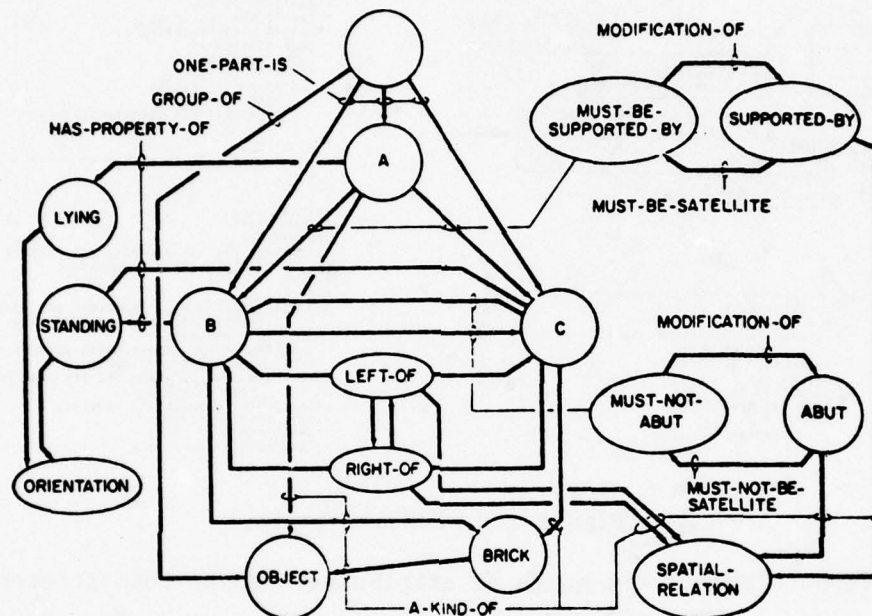


Figure 2.5. Structural description of an ARCH.

of examples of the structure in a network description language. The descriptions included nodes for concepts of physical objects (like BRICKs) in a scene, and labelled links representing physical relationships between the objects (e.g., LEFT-OF, SUPPORTED-BY). The interesting thing about Winston's networks (aside from the fact that he had actually written a program to induce generalizations from them) is that the relationships between concepts could themselves be modified or talked about as concepts. For example, in the very same notation, B could be described as LEFT-OF C, and LEFT-OF described as OPPOSITE RIGHT-OF. Winston also used the same language as his comparison language for determining differences between examples.

One problem with Winston's notation, as with each of the others

mentioned so far, was its complete uniformity. While the notions of superconcept and instance were included in these nets, there was no acknowledgement of their difference from domain-specific notions like location and support. One could not "see" a hierarchy by looking at the structure, and important notions like inheritance were obscured by an overly uniform mixture of domain-specific and general "properties". As I shall contend in later chapters (4, 5, and 8), these are critical drawbacks. However, with the groundwork laid by Quillian, Collins, Carbonell, and Winston, almost all of the semantic net apparatus used in the '70's is already accounted for, and very little has really changed since then.

2.2. Case structures

The work of Chas. Fillmore on linguistic case structure [1968] helped focus network attention onto verbs. Those interested in processing natural language with semantic nets began to think of a sentence as a modality coupled with a proposition, where a modality captured information such as tense, mood, manner, and aspect, and a proposition was a verb and a set of filled-in cases. There were believed to be a reasonably small number of cases (i.e., relationships in which nominals could participate relative to the verb of a sentence), and several people set out to incorporate this belief in network formalisms. The fact that properties in semantic nets were clustered around nodes made the nodes ideal places to anchor cases -- if a node were thought of as a verbal concept, its associated attribute/value pairs could easily be case/filler pairs.

Simmons, et al. [Simmons, Burger & Schwarcz 1968, Simmons and Bruce 1971, Simmons & Slocum 1972, Simmons 1973, Hendrix, Thompson & Slocum 1973] used this notion very early in work that developed from the older "Protosynthes" system. Simmons' networks became centered around verbal

nodes, with pointers labelled with case names to the participants in the action represented by the node (see Fig. 2.6 [Simmons & Bruce 1971, p. 525] -- the verbal node here is C1, a TOKen of the verb, "Make"). The

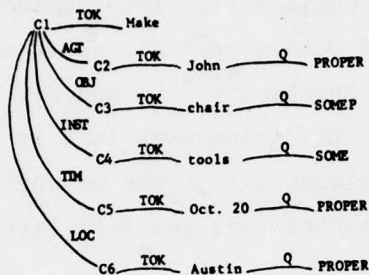


Figure 2.6. A Simmons case structure.

verbs themselves were grouped into "paradigms", according to the sets of case relations in which they participated.

Simmons' networks focused on the understanding and generation of particular sentences -- not much attention seems to have been given in the original work to the semantic network as a hierarchical classification device, nor to the place of general "world knowledge" in the overall scheme. Thus no classification of verbs, or nouns, for that matter, existed outside of the similar case-frame grouping (the paradigms), and no definitions of general concepts seemed to exist at all. Recently, however, some sophistication has been added to these networks, including substantial use of superconcept and "instance" links. In addition, quantification and deductive mechanisms are discussed in [Simmons & Chester 1977].

A similar incorporation of case structures into a network framework was achieved by Rumelhart, Lindsay, and Norman [1972, Norman 1972, 1973, Norman, Rumelhart & the LNR Research Group 1975, Rumelhart and Norman 1973]. Their attempt, spanning several years, included many of the features that Simmons had left out, although their orientation was more psychological and thus dealt with more aspects of memory. The

Section 2.2
Case structures

Rumelhart, *et al.* networks included nodes for concepts, nodes for events, and nodes for episodes -- sequences of events clustered together. General definitions of concepts in the network were encoded in a straightforward manner, with case-like pointers indicating parts of nominal concepts and agents and objects of verbs, as illustrated in Fig. 2.7 [Rumelhart, Lindsay & Norman 1972, p. 224]. Unfortunately, their

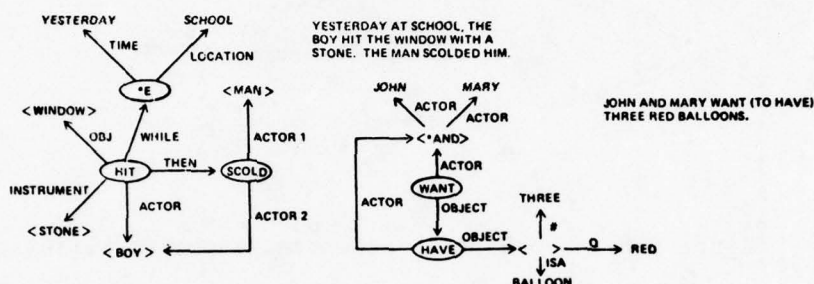


Figure 2.7. Some Rumelhart, *et al.* concepts.

notation was also very uniform, so that all links looked the same. In addition, the infamous "ISA" link (see [Woods 1975a] and [Cercone 1975]) was used to indicate type-token relations as well as subset relations, and many other relations were not motivated or explained -- the English mnemonics are all that we have to indicate their semantics. Relatively little attention was given to the structure at the foundational, logical adequacy level, so that the inheritance relations between concepts were not always clear.

On the other hand, the Rumelhart and Norman group made an effort to account for procedural-type information directly in their notation (using a link called "ISWHEN"), and integrated case-type information with other "world knowledge". They included definitional as well as instantiated (propositional) constructs, and, all in all, they have captured many good ideas in their nets.

Another important piece of work that deserves at least brief mention here is Schank's "conceptual dependency" representation [1972, 1973a, 1973b, Schank, Goldman, Rieger & Riesbeck 1973]. While Schank himself

does not seem to believe in semantic memory [1974, 1975], his conceptualizations very much resemble concepts in systems like Simmons' and Rumelhart and Norman's, as evidenced in Fig. 2.8 [Schank 1973b, p. 201]. A conceptualization consists of a primitive act and some

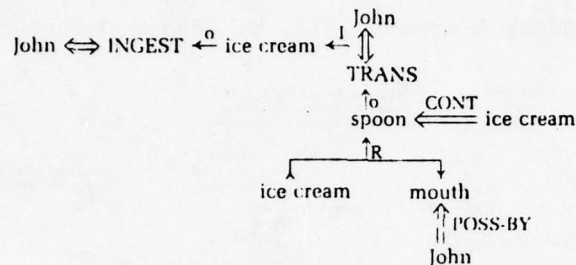


Figure 2.8. A conceptual dependency conceptualization.

associated cases, like "instrument", "direction", etc. In conceptual dependency diagrams, arrows with different shapes and labels indicate the case relations. For example, in Fig. 2.8 the "R" relation (a three-pronged arrow) indicates the recipient case, while the "I" relation indicates the instrument of the conceptualization (one interesting idea that is illustrated here is that the instrument of an action is itself a conceptualization). Each primitive act (e.g., "TRANS", "INGEST") has a particular case structure associated with it, and the higher-level verbs that one sees in the other notations must be broken down into canonical structures of primitives here. Thus, not only does Schank specify a set of primitive relations, he suggests a set of knowledge primitives out of which concepts should be built (this is in contrast to what I shall later refer to as "epistemological primitives", operations for structuring pieces of the representation). Schank's contribution to the study of knowledge representation, while controversial, is an important one. His cases are "deeper" than those of Simmons, and begin to attack knowledge structure at the primitive level. Conceptual dependency was incorporated as the memory structure of the MARGIE system, which was a natural language understanding system that could parse an input sentence into the deep conceptual structure

and rephrase it in a number of different ways. Schank and Rieger [1974] developed some important inferential properties for their memory structures, and their work has had a great influence on much of the later work in the field. The reader should consult [Wilks 1974] and [Cercone 1975] for two excellent expositions of Schank's work.

In more recent work, Rieger has attempted to deal in greater depth with the relations between actions and states [1975, 1976, 1977, Rieger & Grinberg 1977]. "Commonsense Algorithms" (CSA's) capture information of a much more dynamic sort than that handled by the traditional, static concept networks. Rieger has nodes that represent not only primitive actions, but states, statechanges, wants, and "tendencies" (a tendency in CSA representation is a kind of action that takes place without the effort of an intentional force; one such tendency, for example, is gravity). There is a small repertoire of primitive link types which are used to represent the underlying dynamic relationships between the actions, states, etc. ("ten theoretical forms of inter-event causal interaction" [Rieger & Grinberg 1977, p. 250]). CSA links stand for relations like causality, enablement, concurrency, and the like, with the primary emphasis on expressing the cause and effect relationships that make physical systems work. While the notion that causality can be captured in a single link is debatable, CSA's may provide a useful way to express dynamic information that in other systems is supposedly captured by unstructured relational links, and may do so in a complete enough way to allow the simulation of certain physical mechanisms, like the reverse-trap flush toilet [Rieger 1975] and the reasonably complex "Home Gas Forced-Air Furnace" [Rieger & Grinberg 1977].

Two other important treatments of memory with verb-centered case-like systems surfaced in the early '70's. George Heidorn's thesis work [1972] parlayed a simple hierarchical network and instantiation mechanism into a system, called "NLPQ", that could "understand" a queuing problem described to it in English. From this description, NLPQ could produce both an English restatement of the problem and a complete

program (written in GPSS) for simulating the situation described. By including in advance some simple case frame definitions of actions relevant to queuing situations (for example, "unload" takes an Agent, a Goal, a Location, and a Duration), Heidorn provided his system with a built-in definitional context for the description of a particular situation. During an initial conversation with the user, the NLPQ system would build an "internal problem description". This "IPD" comprised a set of instances connected appropriately to the general definitions (see Fig. 2.9 [Heidorn 1974, p. 95]). NLPQ could consult

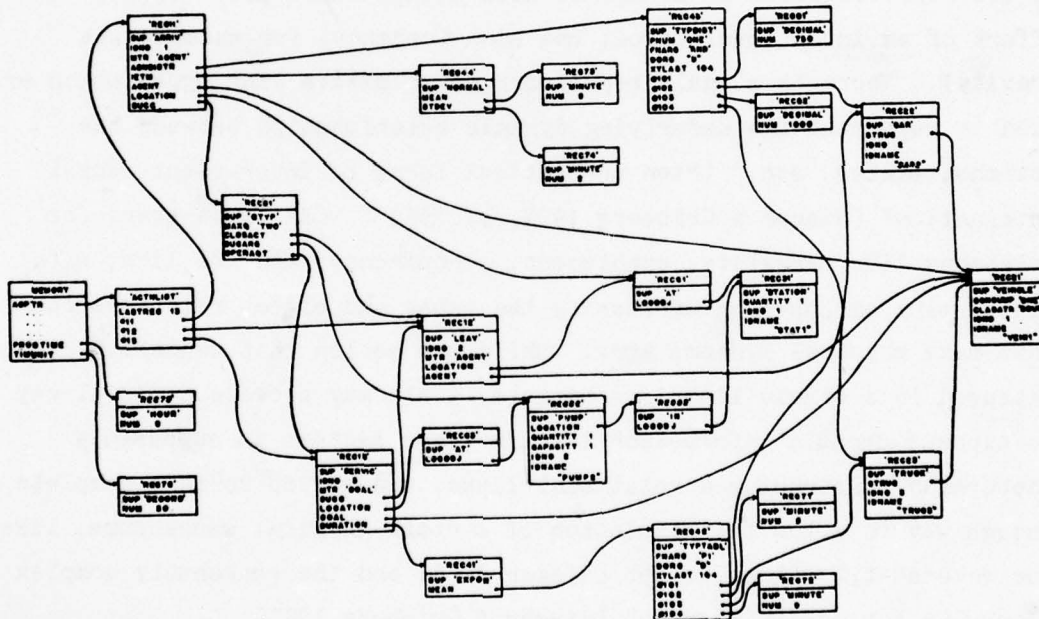


Figure 2.9. Heidorn's "IPD".

those definitions and tell when the problem description was incomplete; it could thus intelligently ask the user for missing information. Although Heidorn's network was very simple-minded and uniform (it was not very deep, concepts had very simple structure, and the "SUP" link was used for both subconcepts and instances), he achieved a rather

dazzling effect by incorporating it in a general grammar-rule language and by starting with a set of concepts well-matched to the simulation language in which the output was produced.

The other "case" study produced a strongly psychologically-oriented memory structure called "HAM" (for "Human Associative Memory") [Anderson & Bower 1973, 1974]. The elements of HAM were propositions, binary trees which represented the underlying structure of sentences. A simple proposition of this sort is depicted in Fig. 2.10 [Anderson & Bower 1973, p. 165]. Relations allowed between nodes in the trees included

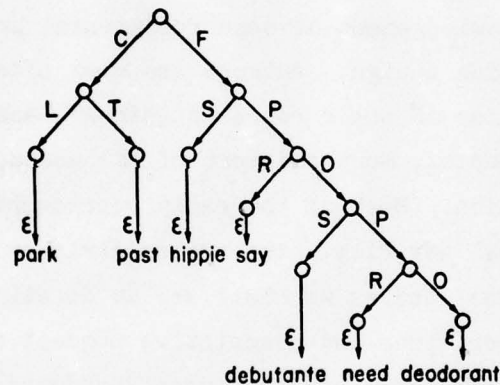


Figure 2.10. A HAM proposition.

set membership (the "e" links in Fig. 2.10) and subset, some cases like subject ("S" in Fig. 2.10), object ("O"), location ("L"), and time ("T"), and some logical indicators like predicate ("P"), "context" ("C"), and "fact" ("F") -- all represented uniformly. Propositions in HAM had truth values, and were supposed to convey assertions about the world; Anderson and Bower's notation failed to account for the internal structure of nominal entities. There were many problems with this simple notation, some of which are discussed in Schubert [1976], a work whose detail on the logical structure of semantic networks in terms of predicates and propositions makes it clear that HAM's propositional

notation is insufficient. However, Anderson and Bower produced an extensive investigation into the state of the relevant philosophical and scientific work at the time of their own work, and their detailed psychological discussions should be consulted. Although their model is admitted to be inadequate and the semantics of their representation is not thoroughly worked out, their book is a milestone of start-to-finish research in a field often plagued by less than thorough work.

2.3. Concern for the foundations

Unfortunately, most of the early work covered above suffers from a lack of explicit acknowledgement of some fundamental principles of knowledge representation design. Authors are most often intuitive when describing the semantics of their representations*, and as the network notations get more complex, more and more of the assumptions are left to the reader's imagination. Most of the early representations were not extensible in a general way (i.e., the system designer must intervene to add new case relations), and as we shall see in detail in Chapter 4, the combination of set operations and descriptive concept operations that the semantic net is based upon has been poorly understood. All of the notations I have mentioned so far are seductively uniform -- conceptual relations (e.g., "agent", "color", "left-of") and underlying knowledge mechanisms (e.g., "superset", "iswhen", "member") are expressed in indistinguishable terms. In Chapters 4 and 5, I will contend that this homogeneity is misguided and confusing.

* For example, "Intuitively, the nodes in the tree represent ideas and the links relations or associations between the ideas" [Anderson & Bower 1973, p. 139]; "In this system a large part of the information is about the words and concepts of the relevant domain of discourse . . ." [Heidorn 1972, p. 35].

Section 2.3 Concern for the foundations

However, in addition to the work described in this report, some recent efforts have set out to remedy this inadequacy. Among the more important of the earlier and concurrent projects that attempted to deal with the expressive inadequacy of semantic nets are the work of Cercone and Schubert at the University of Alberta, and the work of Levesque and Mylopoulos at the University of Toronto, to which I will turn in a moment. Several years earlier, however, Stuart Shapiro [1971a, 1971b] introduced the important distinction between the "item", or conceptual level of network, and the "system" level -- the structural level of interconnection that ties structured assertions of facts to items participating in those facts (i.e., indicates bindings). System relations are the labeled links in the network, and their semantics is determined by the set of processing routines that operate on them. Item relations are concepts which happen to be relational in nature, and are represented by nodes ("items") just as are other, non-relational concepts. Thus, a relationship like "LOVES" would appear not as a link in the net, but as a node. Particular assertions of the relationship would also be nodes, with AGENT and OBJECT system links to nodes for the participants, and a VERB link back to the node for LOVES (see Fig. 2.11 [Shapiro 1971a, p. 43] -- in this figure, the top three nodes are assertions of particular LOVES relationships). Shapiro makes no suggestion as to how the general verb itself should be defined in network terms (that is, what makes a concept LOVES as opposed to any other verb with a similar case frame).

Shapiro's distinction explicitly separates underlying primitive cases from all other, conceptual relations. He also explains how rules for deduction can be encoded directly in his formalism, and discusses at length a language for doing retrieval from his network structure. His early work gives us no guidelines for what the set of system relations should be (his examples suggest linguistic cases), nor does he talk about the semantics of items, except to imply through his search mechanism that sets are important. Shapiro's claim is only that what he

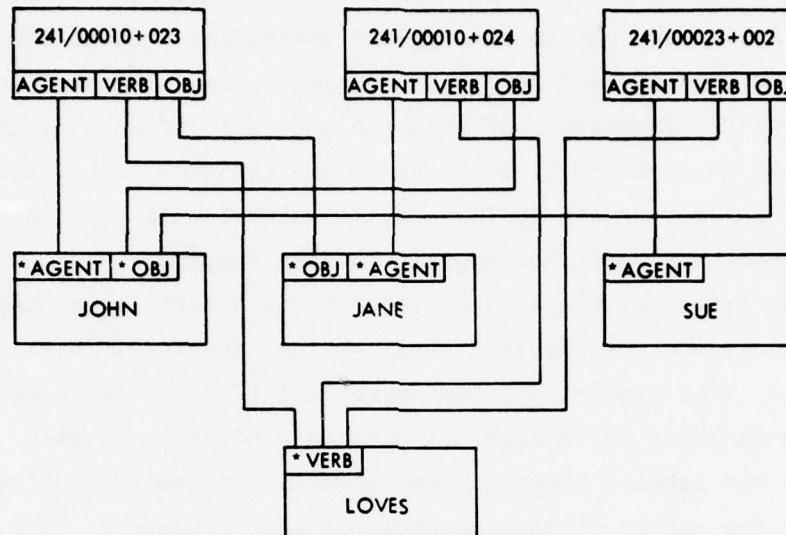


Figure 2.11. Separating system relations from item relations.

has given us is an epistemologically neutral structure, a general language on top of which many models of knowledge might be constructed. This in itself, however, represents a significant advance over previous networks in the distillation of two very different levels of representation.

One of the goals of the work described in this report is to offer a particular set of structuring principles for knowledge to be built on top of a neutral foundation such as Shapiro's. I will, in fact, further differentiate the representation process, producing a third level of representation built out of the neutral primitives of nodes and system links -- it will be this intermediate level that I believe to be the foundation for particular conceptual knowledge.

Between the time of Shapiro's thesis [1971a] and the more recent work to which I have alluded, others have tried to resolve some of the inadequacies of the homogeneous standard evolved from Quillian's Semantic Memory. Hays [1973a, 1973b], in his "cognitive networks", has attempted to differentiate some of the semantics of network notations,

and to be more formal than earlier authors about network structures (he specifies four node types, including "modalities", and five major link types). Among other things, his work has contributed the distinction between a "manifestation" of an object and an "instance"*..

Hendrix [1975a, b, 1976, 1978], in attempting to provide an adequate quantification mechanism for semantic network concepts, introduced what has become a very broadly utilized facility -- "partitions"**, or formal groupings of concept nodes. Figure 2.12 [Hendrix 1975a, p. 239]

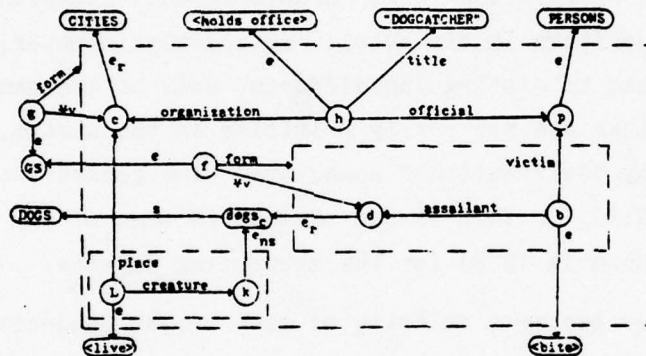


Figure 2.12. A partitioned set of nodes.

illustrates the use of partitions (indicated by rectangular dashed boxes) to represent "Every city has a dogcatcher who has been bitten by every dog in town". In this figure, the two larger "spaces" hold the scopes of the universal quantifiers: the "form" link points to a space representing the scope of the universally quantified variable, which is encoded by a node pointed to by a "for all v" link. The node labeled

* Objects in Hays' epistemology are permanent. However, they do change over time (e.g., a person is at various times an infant, a child, an adolescent, an adult, etc.). Manifestations are different concepts of the same object at different places or stages of its existence.

** Scragg [1975] has, apparently independently, introduced a very similar mechanism, which he calls "planes".

"p" is an implicitly existentially quantified node, representing the particular dogcatcher for any given town.

Partitioning has many potential uses; for example, it can be used to provide a context mechanism, whereby entire areas of memory may be opened up or sealed off at relevant times (this allows reasonable groupings of beliefs). It should be pointed out that the nodes in many of Hendrix's nets represent sets as well as "prototypes", and the introduction of case-like properties for concept nodes makes them susceptible to the same confusions as all of the older, uniform nets (this is evidenced by relations like "creature" and "assailant" being directly encoded as links in his nets). Apparently, however, different space-types are used to distinguish different uses of the same link, and the non-logical links are not really primitive in the system, they're being introduced by "delineations" associated with general verbal concepts like "OWNINGS". This is not obvious in some of the earlier papers, but see [Hendrix 1978] for the supporting details.

Partitions have become a mainstay of many recent semantic nets, and are an indisputably helpful mechanism for representing higher level phenomena like quantification, context, structural "plots" [Grosz 1977], etc. When viewed as a mechanism, with no epistemological claims about their expressive adequacy (which depend on each individual's use of them), partitions do not come under the jurisdiction of the criticisms detailed in Chapter 4. When partitions implement mixed sets of relationships (like "creature" and subset), then they are open to the kind of complaint lodged in that chapter. That is, each partition (space) type used in a system is open to its own epistemological constraints, just as is each use of the simple, general notion of a "node".

In 1975 a very important paper by Wm. Woods appeared; this study of "what's in a link" for the first time seriously challenged the logical adequacy of previous semantic network notations [Woods 1975a]. Woods pointed out the intensional nature of many of the things we call upon

nets to represent (see Chapter 5 of this report), and discussed in detail several important challenges for network notations that had not been previously acknowledged, let alone successfully met. We were asked to begin to consider the semantics of the representation itself, and to be held accountable for things previously brushed aside under the auspices of "intuition". The work described in this report is to some extent a broader and deeper investigation in the same spirit as the Woods paper, a continuation of the semantic investigative work only begun there. It is hoped that many of Woods' challenges have been overcome by the structures illustrated in later chapters.

Some of the issues raised by Woods -- the more logically oriented ones -- have been recently treated in a series of papers by Cercone and Schubert [1975, Cercone 1975, Schubert 1976]. In their attempts to extend the expressive power of network notation, Schubert and Cercone have expended considerable effort in the investigation of the underlying logical content of the node-plus-link formalism. Many of the issues of knowledge representation that are emphasized in this report were raised in various papers from Alberta; in particular, an excellent criticism of the naive notion of the existence of a small number of "conceptually primitive relations" (i.e., cases) reflects a similar intuition about roles, to be developed in Section 5.1.3.1 (see [Schubert 1976, pp. 168-170], and [Cercone 1975, pp. 79-80]).

The notation developed by Schubert and Cercone is propositional -- an important basic node type in the network is the predicative concept node, which is instantiated by conjoining at a proposition node a pointer to the predicate and a pointer to each argument of the predicate (see Fig. 2.13 [Cercone 1975, p. 36]). The links used are all predefined system links, used only to point out the particular predicate invoked and to order the arguments. All of the conceptual work is done by the particular predicates pointed to with "PRED" links from the proposition nodes. Schubert and Cercone claim also to have concept nodes for individuals and sets, although it is not clear from the

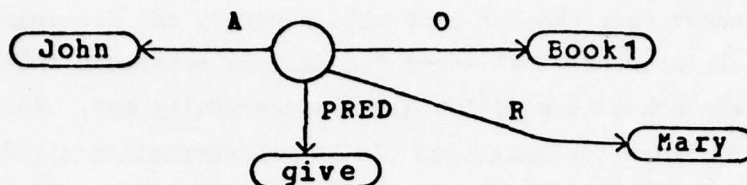


Figure 2.13. A proposition node.

notation where these interpretations are expected. Given the propositional nature of the notation, a series of logical connectives and quantification conventions can be unambiguously (and explicitly) represented. In addition, Schubert and Cercone provide facilities for lambda-abstraction and various other intensional operations, and include time primitives for certain types of predicates. Schubert [1976] discusses the clear correspondence of his notation to predicate calculus, providing for the first time a clear standard of reference for network (logical) adequacy*.

While the work of Cercone and Schubert begins to answer some of the questions raised in Woods' paper, theirs is still only a neutral logical language. This notation, as all others discussed so far, offers no guidelines to its users on how to structure concepts in terms of the primitives of the notation. The language is as general, uniform, and low-level as predicate calculus and it is up to the designer of the particular network how to structure his world in terms of predicates and propositions. While Schubert's notation unambiguously accounts for many of the underlying logical operations of the semantic network, something more seems to be needed for it to be a truly useful representation of knowledge. This seems to involve looking at network structures at a slightly "higher" level, and I pursue this in depth in this report.

* See also [Simmons & Bruce 1971] and [Hendrix 1975a] for earlier discussions of the correspondence between semantic nets and predicate calculus.

Section 2.3 Concern for the foundations

Some hints on higher level primitives have been afforded us by some more recent efforts in network formalisms. Fahlman [1977] has designed a network system comprising two major parts: a parallel memory scheme, which allows propagation of markers through a network composed of special-purpose hardware; and a language (called NETL) for representing knowledge on top of the parallel memory. There are several important things to note about Fahlman's work. His is perhaps the first attempt to account for network-implementing hardware in its own right. The marker propagation and detection mechanism eliminates much of the costly search intrinsic to previous, non-parallel systems. Further, he introduces the idea of a "virtual copy" as a dominant organizing concept. This is a convenient way to think about inheritance in semantic nets, since it lets us assume that all properties at a parent node are (virtually) available at its subnodes. When a real copy is needed, as, for instance, when a property is to be explicitly modified, Fahlman has us create a "MAP-node". The parallel-processing scheme makes virtual copy and map links act as short-circuits in the appropriate circumstances, thereby allowing any inherited definitions to be immediately available.

Further, Fahlman introduces the "role" as a type of individual, whose universe of existence is another concept. While he at times, I believe, confuses the notion of a functional role (like "AGENT") with that of a role filler (like "PERSON"), he seems to be on the right track in terms of the structure of concepts. In the work reported here (see Chapter 4), this role notion has been found to be critical, and SI-Nets have what amount to MAP-nodes also. A good deal of Fahlman's foundations could be used to support other network schemes.

"Role-nodes" as parts of structured descriptions also constitute a critical element in the work of Philip Hayes [1977a,b]. Hayes' networks have two levels of structure, just as those to be presented in Chapters 4 and 5 have; the internal structure of "depictions" (concepts), and relationships between depictions as wholes. Briefly, a depiction

expresses the structure of an entity through a set of PARTOF and CONNECTED relationships between other entities that make up its parts. For example, in Fig. 2.14 [Philip Hayes 1977a, p.93], the depiction

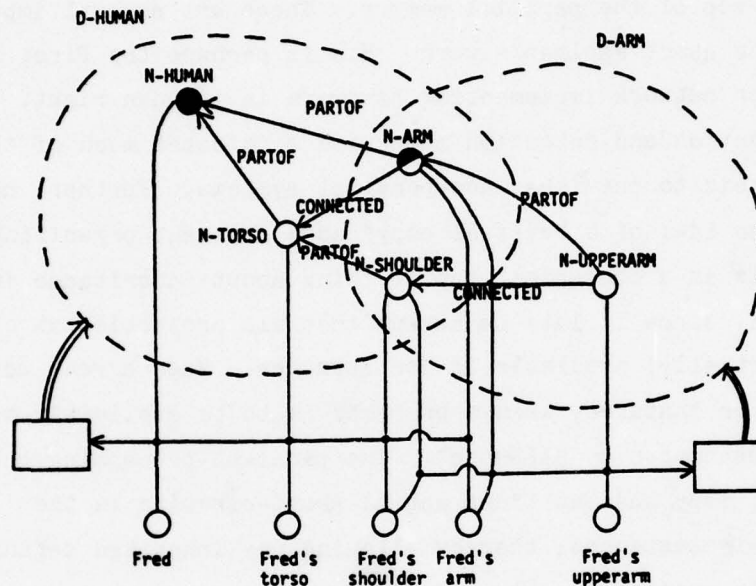


Figure 2.14. Hayes' depictions and binders.

"D-HUMAN" (indicated by dotted lines) partially expresses the structure of a human (represented by the node, N-HUMAN) in terms of an ARM and a TORSO. In the depiction, D-HUMAN, N-ARM acts as a depicter; at the same time, in D-ARM, N-ARM is the depictee - the subject of the depiction*. Thus, while it is a thing unto itself in one structure, it acts as the specifier of a role to be filled in another. In some cases, Hayes contends (and I concur), the role can only exist within the larger context. For example, an arm cannot exist without implying the

* While N-ARM is the same node in both depictions, links to it are only "visible" from the depiction from which it is viewed. That way various uses of ARM from more than one context can be kept distinct.

existence of some human; in that case, N-ARM would be an "SQNODE" for D-HUMAN, and the dependency would be expressed in an "SQN structure" (for sine qua non) involving D-ARM and D-HUMAN.

While Hayes does not distinguish the role itself from the role filler (see Chapter 4), and "CONNECTED" is much too simplistic to capture relations between roles, the very fact that Hayes has roles at all is significant. Concept structure involving roles is strictly enforced in instantiation, using a structure called a "binder". In Fig. 2.14, there are two binders (indicated by the rectangular boxes, the arrows coming in to them, and the dots at intersections), representing "Fred" and "Fred's arm". The binder captures the fact that roles are inherited as part of a structure. There are explicit connections between role definitions (in the depictions) and role filler/instance pairs (in the binders), just as I propose in Chapter 5 (although the exact nature of the relationships is not spelled out in Hayes' structure). The explicit acknowledgement of these relationships is a very important development in the history of semantic networks.

Finally, a joint concern for higher-level (non-logical) structures and their semantics in a semantic network formalism has surfaced in the work of Levesque and Mylopoulos at Toronto [1978, Levesque 1977]. Their efforts attempt to provide a procedural semantics for the relations in a network by associating with a class (concept) a set of four operations: add an instance, remove an instance, fetch all instances of the class and test for being an instance of the class. Classes are given internal structure with slots; parts fill these slots, generating a "PARTOF hierarchy". The classes themselves are organized in an "ISA hierarchy", which expresses generalization relationships between classes and subclasses.

In addition to these two hierarchies, the system of Levesque and Mylopoulos also has an "instance hierarchy". Every class is itself an instance of the class, "CLASS," which is termed a "metaclass". Adding this distinction allows a precise account of inheritance, and of

relations often mistaken in more uniform schemes -- including the descriptions of the programs themselves. Levesque and Mylopoulos also provide nice accounts of the distinctions between structural and assertional properties and between property attributes and property values, and account with their procedures for the interdependencies between pieces of a structure. As such, their account would provide a good set of tools for exploring the semantics of the representation to be presented in Chapters 4 and 5. The only major shortcoming is the lack of an explicit representation of the relationships between the parts of a class, since their dependencies are only implicitly accounted for in the four programs associated with a class definition.

Chapter 3. Some Methodological Points, and Two Domains

Scientific research is most often judged by its results. A proven theorem or a newly synthesized molecule are very visible parts of successful research programs. However, work in a young or rapidly changing area rarely culminates in such clear-cut endproducts. Rather, germs of potentially fertile ideas are often hidden within groping attempts to solve broadly-defined research problems. The "results" of work in which the majority of time is spent in trying to find the right questions to be asked (and in which the answers that follow seem obvious and almost trivial) are buried in the development of those questions. Thus, good ideas in these areas are very much at the mercy of methodology -- a possibly seminal idea can be thoroughly obscured by a confused approach.

Unfortunately, this seems to have been the case with semantic networks. To produce from such an intuitive, vaguely-defined notion as "associative semantic representation" a clear and useful kernel of ideas, a firm technical foundation and a rigorous research discipline are needed. One cannot simply start trying to "put natural language in a semantic net"; it is not even clear what is meant by the term*. (As we shall see in Chapter 4, the most common notation is so seductive in its uniformity as to make all of the "obvious" representations ambiguous or inadequate.) Yet people have tried repeatedly, each time developing a new notation and each time failing to appreciate fundamental methodological principles. One of the aims of this report is to help

* See [Brachman 1978] for an analysis of five different kinds of "semantic" net.

make clear why other attempts have failed to achieve their elusive goals, in the belief that one way to better understand research on representations of knowledge is to examine the approaches taken by researchers toward their problems.

In this chapter, I outline some of the important questions that one should be aware of before he begins research on semantic networks. In particular, the intent is to understand why the most common foundations for semantic nets are inadequate, and what it will take to provide an adequate foundation. This chapter suggests an approach to the foundations issue, which I will attempt to follow in the rest of this report. Here I will discuss briefly the importance of being aware of implicit assumptions and expectations about a representation scheme, of developing an epistemological foundation for the representation, and of choosing an appropriate domain to represent.

3.1. Assumptions and expectations

As we saw in Chapter 2, the last ten years has produced a great many ambitious projects built around "semantic networks". Quillian originally expected his nets to "allow representation of anything that can be stated in natural language" [1969, p. 460], and more recent attempts have taken off from there. For example, nets have been invoked to capture "meanings of sentences", "actions", "events", "facts", "properties", "assertions", "objects", "relations", "expressions", and most pervasively, "concepts". Yet rarely can we find a precise definition of what these things are that networks are expected to represent. Most often, authors rely on their readers' intuitions about things like meanings, concepts, facts, and properties, failing to set down in clear terms what behaviors to expect of each of these entities. Thus we are left with no standard by which to judge the success or failure of a representation. Only very recently has the issue of

logical adequacy for networks been raised [Woods 1975a, Schubert 1976, Brachman 1977]. To make the issue of logical adequacy -- or any other kind of adequacy [Brachman 1978] -- even meaningful, we require precision about the target of the representation. What do we expect these representations to represent? One of the fundamental questions that I seek to answer in this paper is, what, really, is a "concept"?

Beyond the failure to be clear about the objects of the domain, semantic net research generally suffers an additional methodological failure. Virtually all of the "standard" net representations seem to have been born of the assumption that it is adequate to represent a relationship by a link*. One simply makes a link type for each relationship to be expressed in the knowledge base; nodes for entities are simply conglomerations of such associations in which the entities participate. The result is a completely uniform-looking net which is supposed to express relationships of many different kinds (provided, of course, that one has been precise about what those relationships are). A great deal of faith is placed in the adequacy of the intuitive representation.

Unfortunately, assumptions about the differences in the nature of links are never explicitly expressed in the representation itself. The meanings of links exist only in the processing routines created to manipulate the structure. There are at least three serious implications of the failure to be precise about the import of links (i.e., the failure to represent the meanings of conceptual relationships in the network itself): 1) nets are not easily extensible or alterable -- new

* Shapiro [1971a], however, draws a sharp distinction between two different types of relations -- "item" relations for conceptual relations and "system" relations for underlying primitive relations. Schubert [1976] deals with relations explicitly as logical predicates, and is very clear on the semantics of his notation. However, neither of these authors gives us a methodology for encoding things like "meanings" in his representation (although both papers are sprinkled with examples). That is, neither offers what I shall call an "epistemology".

routines must be added to handle new links or concepts, and old routines must be altered to handle changes in the meanings of relationships; 2) links easily fall into ambiguous uses (see Section 4.2.4 for details), and improper interpretation of a link is easy, since all one has to go by is a name; and 3) it is impossible to tell if the representation of an object in the domain has been accurately carried out. It is critical to be precise about how a representation scheme (language) is to capture an entity from the domain (i.e., what the primitives of the language will be taken to mean, and therefore what pieces of the domain they can stand for). Failure to be precise about the primitives of the representation language means that the semantics of the notation itself is not well-defined.

In Chapter 4 I will try to reverse these methodological trends. I will attempt to make explicit certain assumptions about network notation, in order to understand more precisely what the representation is expected to represent, and how it is to do that. I will investigate in depth what a "concept" is, and in Chapter 5 will settle on a well-known philosophical construct (intension) to help make the representation more precise. In addition, I will attempt to find a way to keep the meanings of conceptual relationships in the network itself, thereby avoiding the above-mentioned problems. No matter what vague kinds of things we would like networks to represent, they must have precise and singular interpretations for the routines that process them.

3.2. Reflecting underlying operations -- Epistemology

The basic language of semantic networks is very simple and general. Any entity that we wish to be able to talk about is represented by a node, and all of the relationships in which the entity participates are indicated by links attached to that node. There is no limitation on what can be considered an "entity" -- relationships themselves can be

nodes in most semantic network schemes.

The very general languages passed down to us from those projects cited in Chapter 2 do not offer us very much guidance as to how we might break down our own particular domains into nodes and links. It is up to the network designer to choose an appropriate set of conventions for relationships and entities, and then to embody those conventions in the actual network he builds. This is very much like being given a general programming language (like LISP) and a particular task, and being asked to write a program to handle the task. What data objects and what routines to create are totally up to the programmer, and there is nothing in the language that tells him how to write a program and have it consistently and meaningfully operate on its data.

In both of these cases, the implementer is given a set of constructs that presumably can handle any task -- nodes and links (in the first case) or function calls (in the second). He just (!) has to find the right way to implement his own system. The languages impose no "worldview" on the user; he has only a completely general set of primitives in which he must directly encode his domain.

While LISP and semantic networks are very powerful in their generality, there is a problem in their presenting the user nothing more than a handful of primitives -- particularly in the case of networks*. The user has a set of concepts and relationships that he wishes to express in a network, and therefore encodes these directly as nodes and links. In addition, he invariably believes that, by virtue of their being "concepts", these things should exhibit certain characteristics true of concepts in general (for example, they have "instances"; more

* We generally have had at least some experience in programming which has taught us to think in terms of levels of abstraction that impose a higher-level structure on the implemented code -- in addition, we can embed function calls within function definitions to reflect this structure. This experience is lacking for network representation.

general concepts pass properties to more specific ones; etc.). Yet there is no way in the notation that he can express distinctly these two very different sorts of abstraction -- he must encode features of concepts as notational objects directly as nodes and links, too. There is no separation between an operation that some concept participates in because it is represented as a "concept node" (e.g., "the concept TELEPHONE has three instances") and a property of the thing represented by that concept that is true by virtue of its place in the domain (e.g., "telephones are black").

That is, the structure of representations for "concepts", "instances", and "properties" is a different sort of thing than the structure of actions and events and objects. Yet the uniformity of the semantic net forces the former to be obscured by the direct encoding of the latter. Look at virtually any network in the literature, and you will find that the primitive links of the net include relationships from the domain being represented.

What is needed is a separation of operations on concepts as formal objects in a representation from high-level domain-dependent relationships to be expressed between elements of the domain itself. Semantic nets need to offer their implementers an epistemology, a set of primitive structures for encoding knowledge and rules for combining those structures into well-formed representations of individuals and classes of individuals. Operations like concept definition, individuation of a description, and property inheritance are the fundamental epistemological mechanisms of this type of representation of knowledge, and conceptual relationships like "COLOR", "AGENT", etc., should be expressed in terms of these primitives, and not directly in nodes and links.

Chapters 4 and 5 will show how a detailed analysis of what the primitives (in this case, nodes and links) are supposed to represent can bring out the important operations underlying a knowledge representation. I will illustrate how standard network schemes fail to

distinguish between the foundational, or "epistemological", level of knowledge representation and the domain-dependent, "conceptual" level*. The approach here will instead be to take each epistemological operation and make it into a primitive available to the user, thus producing an epistemologically explicit representation. This is a language not of uniform nodes and links, but of several different types of nodes, a fixed set of known links, and a set of rules for creating formal concepts out of such primitives. As we shall see, adding an epistemology to a general representation language (i.e., building a language at this level) enhances the perspicuity of structures built in the language, and creates a well-formedness criterion for such structures. In addition, the intermediate level of epistemological primitive allows us to write completely general, domain-independent routines for building, extending, and using networks.

3.3. Domains

It is very difficult to appreciate the representational adequacy and expressive power of a representation of knowledge in isolation. While one may easily generate purely abstract hypothetical "examples", the import of the representation is not apparent until it is applied to real-world problems of some depth. Whether one starts with a domain of study and determines from it a set of relevant problems, or he starts with a set of issues and finds a domain which exhibits the desired behaviors, it is the domain through which the examples become meaningful and the model becomes convincing. This being the case, it is imperative

* See [Landsbergen 1976] for a brief discussion of a similar distinction. The PHLIQA1 system described in that paper distinguishes between "formal" (what I call "epistemological") semantics and "referential" ("conceptual") semantics. In addition, there are other levels of abstraction that we could use to analyze networks, as hinted in Chapter 2. See [Brachman 1978].

that the important problems clearly and readily surface from the domain.

Here I would like to emphasize the importance of a carefully thought out choice of domain. Many worlds are appealing in their simplicity and the fact that the important problems may be obvious just from looking at those worlds. But, as I have mentioned, things may appear simple only before we carefully analyze the objects of the domain that are to be represented. And not only is the overall choice important -- it is critical to try to represent the most difficult and subtle problems of the particular domain. The "real" problems must be faced before a representation that claims to handle a relatively simple surface phenomenon can be claimed a more general mechanism.

This has been a stumblingblock for semantic networks. Most representations can in some sense handle English sentences centered around verbs, and to some extent, representations for verbs themselves. But none of these representations are adequate to express the structure of nominals, or the more subtle intensional operations of natural language (e.g., relative clauses -- see [Woods 1975a, pp. 60-65] -- and "meta-description" -- see [Smith 1978]). Thus, any claim about a representation "handling" natural language is dubious.

Once a domain that is a rich and natural source of problems is found, it may itself be too broad for study in a single research project. A common methodological trait of recent work in Artificial Intelligence has been the limitation of the domain of study. When developing a natural language understanding program, a CAI system, or a representational structure for knowledge, it is important to be able to focus on the kernel set of problems without having to deal with an insurmountable supply of extraneous difficulties. It may be overwhelmingly difficult just to find the important and interesting questions when an overly broad domain keeps offering digressions.

Yet while it is easy to justify limiting one's domain, it may also be easy to simplify away the important issues. It is important to

Section 3.3 Domains

account for not only one's domain as a whole, but for any limitations to be placed on it. The important theoretical problems must be preserved through any simplification of the target subject matter.

The semantic net is no exception here. Nets are almost always justified in opening paragraphs as candidates for "representing knowledge". Yet the examples treated in the papers almost always cast aside the more subtle parts of knowledge that might provide true tests of the adequacy of the representation (e.g., the apparent ease of making STRAYDOGS a subconcept of DOGS in [Hendrix 1975b], the simplistic examples like "Peter put the package on the table" of [Norman 1972], etc.). Here I will try to overcome another methodological problem in semantic net research by choosing two realistic and complex domains of knowledge for study. One -- the understanding of the structure of a particular interactive computer program -- is limited in scope, but is still a source of deep intensional representation problems. The other -- a natural language information consultant -- is more like the typical domain, but can be limited to the understanding of English nominal compounds without loss of the key issues. Both domains are rich in representational problems, and deep in their structure. While disparate, they possess a common core of requirements for a representation, and a representation that might adequately handle both would be powerful indeed.

3.3.1. A document information consultant

Consider the following scenario, in which we have each probably participated many times: I wish to begin a new project assigned for a course, say a research paper on semantic networks. Not knowing where to begin in the literature, I approach the professor who assigned the project, and ask, "I'd like to do my paper on the logical adequacy of semantic networks -- what references can you recommend?" My mentor contemplates for a moment, and commences enumerating a reading list that

covers all of the major work in the area: "Read Quillian, of course, and don't forget the 'TLC' paper. Carbonell's work extended that, so check his papers, especially 'AI in CAI'. Speaking of CAI, Brown and Burton's SOPHIE system used a network for representing a circuit. And read the recent work by Schubert; and don't forget Shapiro's thesis. And you should probably check into the new papers on KRL and frames -- they are both languages somewhat similar to semantic networks." I might reply with "Didn't you mention in class a paper by Simmons in the Rustin book?", and be told in return, "Oh, you mean the Schank and Colby book -- you might read that, but I don't believe that he has much to say about logical adequacy."

What would it take to have the same dialogue with a computer? How hard would it be for an automated assistant to produce such an annotated reading list? Let us imagine an automated document consultant, a program that has a collection of knowledge about documents in a given area and that might be queried in a natural way for groups of documents about the topics in which we are interested (Woods discusses a more general class of these memory extension devices, which he calls "Mnemo" machines, in [Woods 1975b]). What characteristics of the above-sketched interaction with the professor are relevant to such a system?

First, it should be clear to anyone who has ever been involved in such a dialogue that the consultant has a much broader knowledge base than just the topic with which he has been queried. He cannot be expected to have in the forefront of his mind complete descriptions of every article that he has ever read relative to my query -- instead, he no doubt has an extensive, well-organized familiarity with the entire area of knowledge representation, and remembers a small number of important features about each of the references he has read (many of which will take prompting with some related topic to bring to consciousness). This latter should be true for two reasons: 1) the number of documents read by such a professor is bound to preclude detailed knowledge of each -- there is just too much information

available to stay on top of all current work; and 2) as time passes between readings, much information is lost, and only prominent features remain. Memories tend to become stereotypical, and access paths get obscured*.

In addition, the requests inevitably vary from the way that the information was first stored in memory. I might have said "associative memory nets", or "foundations" instead of "logical adequacy". Further, these phrases might never before have been encountered together by the consultant. Yet the results would have been the same. An important property of this kind of consulting is the way that "conceptually different ways of expressing the same fact are all acceptable and understandable to the system" [Woods 1975b, p. 1].

As we note from the hypothetical dialogue, one reference very often leads to another. Associations of many sorts in the reader's memory connect many documents about similar topics (e.g., contrast the transition from Quillian to Carbonell -- an historical progression -- with moving from CAI to SOPHIE -- a topical association -- and finally with the way in which "Schubert" can lead to "Shapiro"). In fact, not only is knowledge of a document connected to knowledge of other documents, it must be associated in many ways with the person's knowledge in general, or it could not be retrieved from the query. Having "understood" an article implies having tied the concepts presented there to things already known (and having created new structures out of old concepts).

It is the thesis here, then, that to create a computer program capable of providing a literature consulting service like the one that a professor might offer, we need much more than a simple topic index of the kind characteristic of a typical "information retrieval" system.

* See [Bartlett 1967] for some interesting early thoughts on these features of human memory.

Required instead is at least a broad knowledge of the concepts of a topic area, a facility for incorporating new conceptual citations for references into that knowledge base (in lieu of actually reading the documents and deriving the same kind of understanding that a human would), and an associative access mechanism for retrieving items relevant to a query.

This is a tough bill to fill at the current stage of our knowledge, so I will attack the document consulting problem by focusing on the heart of such a system -- the organization of the memory of such an "automated professor". Without a structure for both the general conceptual knowledge and the document citations, one which would allow associative retrieval operations and assimilation of new information, we could not even begin to consider an entire system.

A common device that one might use to begin investigating this domain is the annotation. An annotated bibliography captures the kind of digested and assimilated outline of documents postulated to be at a professor's disposal, and would make an ideal input for a program well versed in a general area but not familiar with any particular parts of the literature. The annotations* would produce new interconnections between concepts already present in memory. So let us consider as our first domain the investigation of a knowledge representation that might handle general concepts and the assimilation of annotations that make reference to those known concepts. Despite the disadvantages outlined above, the associative nature of the semantic network makes it a good candidate for this task -- provided that we can construct a version of the formalism that stands up to the challenges to be discussed below.

* For example, "Semantic net research at BBN, specifically the SCHOLAR project, dealing with 'natural' kinds of inferences," "A short summary of a natural language project at Rutgers," "Presents a new view of the segmentation of human memory (surface, shallow, and deep memory), and provides linguistic evidence in its support." See [Brachman 1973] and Section 6.1 for more examples.

Thus, our main task is to make some form of associative network support the representation of the kind of document descriptions that we find in annotations.

There is one further limitation that we are forced to make, and I spend the remainder of this section discussing its implications for a network representation of knowledge. A look at some annotated bibliographies (see [Brachman 1973] and Section 6.1, for example) shows that the telegraphic style that one uses to concisely express his feelings about a document makes heavy use of a common English information compaction device -- nominal compounding. This linguistic device allows one to create rather freely new terms, by juxtaposing two already well-known terms and treating the result as a single unit. The first is taken as a modifier of the second, and the modification that has been abbreviated is generally obvious. For example, rather than say "the science of computers", we will invariably say "computer science" (this works recursively as well, allowing compounds to be built from already compounded subunits, e.g., "computer science technology"). Compounding is a tremendous space-saver, and is extremely common.

Nominal compounding is something that we do so easily that we rarely pay attention to its overwhelming productivity. We are also rarely aware (except in the case of a newly generated compound that we fail to understand) of the amount of conceptual processing necessary to understand noun-noun compounds. The fact that the conceptual relationships that bind a compound together are not usually explicit never comes to our attention. While many compounds have a nominalized verb which at least suggests the relationship between the two words*, some of the most productive methods for turning new phrases yield compounds like "apple core", "hydrogen bomb", "automobile plant", "blood

* Here I deal only with two-word compounds, although the process is recursive. The parsing problems that exist with multi-word compounds are extraneous to the discussion here.

vessel", "swan boat", "oil slick", "bull ring", "station wagon", "baseball season", and "document information". Such noun-noun compounds express a virtually infinite variety of subtly different relationships, with no indication at all from surface structure as to the nature of those relationships. Unfortunately, not much is known about how we might implement a program to emulate our own ease in understanding compounds.

A great deal of insight into the generative aspect of these linguistic tricks was gained by Robert Lees, when, in 1960, he produced a comprehensive work on nominalization in English. Lees discussed in detail the transformation of verbals to nominals, and analyzed the generation of nominal compounds like those mentioned above (most of those are from his book). His analysis broke noun-noun compounds into ten classes, each being characterized by the underlying basic grammatical relations between the elements (this was assuming that some verbal relationship, ultimately expressible in a sentence, could be found between the words; some examples of his classes are Verb-Object, Subject-Verb, Object-Prepositional Object, etc.). For our purposes here, however, Lees' work is of little theoretical help. His account is specified in now obsolete transformational terms, and is a purely generative description of compounding. All of the syntactic information is lost in the transformations, and thus we have no assistance from his account on how to comprehend the compounds. Syntax would be of very little help here, anyway, since the underlying relationships between the terms are dependent on the terms themselves, and we would still have to rely on conceptual information to tell us whether any connecting relationship posited in a deep structure would be appropriate or not.

So the central capability that the network notation must support is the mechanical understanding of the compound expressions that form such a large part of the annotated bibliography vocabulary. If the device were not so productive, one might consider making such phrases lexical units, as is commonly done in natural language understanding systems.

But their infinite variety presents the same conceptual demands as does the more general case of sentence understanding. In addition, nominal compounding demands some particularly powerful capabilities of the notation in which the meanings of compounds are to be expressed:

- 1) Compounds are tremendously ambiguous -- the same two-word expression can indicate several potential relationships between the constituents (for instance, the phrase "woman doctor" has two very different interpretations). Thus a notation is required to be able to represent adequately all alternatives, and facilitate the different inferences to be drawn in those independent cases.
- 2) If asked to explain a phrase like "lion house", one might offer, "a house for a lion", "a house belonging to a lion", "a house that a lion lives in", or "a house suitable for lions" [Gleitman & Gleitman 1970, p. 95]. That is, one would make use of the particular subset of his own conceptual repertoire that was relevant to expressing his particular interpretation of the compound. A representational mechanism should be responsible for the stringing together of currently available concepts (not universally "primitive" ones) to make up a new definition; that is, it should allow idiosyncratic definitions. In addition, if some of these concepts are themselves vaguely defined (see Section 6.4.3), then the new concept will inherit that vagueness -- that is, the representation should allow structures for vague ideas.
- 3) On the other hand, all of the above interpretations of "lion house" are basically the same. (One certainly would allow those as reasonable paraphrases presented to a system over a long period of time.) Thus, a representation must facilitate the determining of paraphrase relationships between seemingly different interpretations of compounds.
- 4) As stated above, many types of compounds do not explicitly indicate the underlying relationships between their constituents. Thus a mechanism must be available by which one could infer a reasonable relationship between any two terms that could be meaningfully compounded. This is very similar to the problem of paraphrase retrieval. (Contrast this with the general case of sentence understanding -- normally, we are at least given a verb on which to base relationships; here there is often no indication of a reasonable verb.)
- 5) A study of compounding reveals that many similarities exist between nominal and verbal elements. Reasonable interpretations can be found for certain classes of compounds if nominal constituents are afforded case structures similar to those usually given to verbs

(see [Chomsky 1970], and Chapter 6). Thus, the important relationships that exist between verbs and nouns derived from verbs (i.e., nominalizations) should be expressible in any notation that purports to be adequate to represent compounds.

In Chapter 6 I will investigate some of the implications of these requirements for formalisms like semantic nets. As I aim towards that goal, and develop a new representation scheme in Chapters 4 and 5, I will bear in mind that the domain of nominal compounds requires a very general representation that is fundamentally productive, and that facilitates paraphrase, inference, and analogy on all of the relationships that it represents.

3.3.2. Understanding Hermes

The other area to which I will address myself is more limited in its scope. While bibliography annotations might range over a great many subjects (depending, of course, on the content of the references), the range of subjects arising out of the single computer program, "Hermes" [Myer, Mooers & Stevens 1977], is much narrower. Yet while it covers only a limited area, knowledge about this program involves subtle, interrelated definitions of highly structured objects and routines. In this section I will introduce some of the salient features of the Hermes program, and see why it is that we might want to develop a representation for encoding knowledge about that program.

Hermes is a large and sophisticated interactive program that is currently being used to read, write, and process electronic "mail" routed through the ARPA computer network. A user logs into his host computer, invokes the Hermes program, and then issues commands to manipulate the message environment. When the user terminates a command, Hermes will carry out the specified processing, and subsequently return to the user for the next command.

Section 3.3.2 Understanding Hermes

The object of primary concern to Hermes is the message. A message can be any string of characters sent from one host computer on the network to another. However, by convention, there is a structure that the Hermes program will try to impose on messages -- the system interprets the text as an ordered set of message fields, where a field is a label (ended with ": ") followed by some structured contents. For instance, there is a field to indicate the recipients of the message (with label, "TO: ") whose contents is a set of legal ARPANet directory names (e.g., "TO: MYER@BBNA, RBRACHMAN, BURTON@BBN-TENEXD"). Messages themselves reside in message files. For each user a special message file, his "inbox", is maintained as a repository for incoming messages. Messages coming into the system are automatically dropped into these inboxes, regardless of whether or not the user is logged onto the computer.

The set of commands available to the Hermes user is extensive. Hermes commands exist for examining message files (GET), directing attention to a particular subset of the messages in the file (such a subset is called a "message sequence"), and for printing (PRINT, TRANSCRIBE, SURVEY, etc.), listing (on a line printer -- LIST), and filing (FILE, MOVE) these subsets. The user can also create an outgoing ("draft") message by building and editing the desired fields of that message; Hermes has the facility to prompt the user for these fields, or he can initiate their creation himself. In addition the user can do sophisticated searching using objects called "filters", he can do formatted output and "template"-controlled message creation, he can reconfigure existing messages (EXPLODE), and he can have the system automatically respond to (REPLY) or forward (FORWARD) a received message.

To help alleviate the burdensome first impression one tends to get of the system in all its glory, Hermes has been fitted with a set of defaults that allow the user to specify as little information as possible and have the "right" thing happen. Yet this, too, adds

complication to the overall system: the user now has at some point to contend with the "switches", which record the default settings. While the system is tailored to appear simple, it is nevertheless rarely obvious to the naive user what he is supposed to do to accomplish his objectives. The system designers have tried to make a command's name suggestive of its function. However, it would be impossible to anticipate in advance all possible users' conceptions of the Hermes world, and after all, each command has only a single, brief name. To further complicate matters, many of the commands are only subtly different -- how is the naive user to know if his own objective is to PRINT or TRANSCRIBE, or to SURVEY or SUMMARIZE a message?

While the system is equipped with several types of automatic documentation aid, Hermes itself cannot answer questions. If a user wishes to know "What is the difference between SURVEY and SUMMARIZE?" or "How do I read my mail?" he must ask one of the resident human Hermes experts (assuming that there is one resident at the user's location). The second application task for the knowledge representation, then, is to support an automated consultant that would assist the user in learning about Hermes.

An intelligent Hermes consultant that would answer questions like the above would be a great asset to users of the system. If a command were spotted that might be of use, the tentative user might ask, "What does the SURVEY command do?" and expect a reasonable explanation. He then could ask, "How do I use it?" to learn how to use the command. One of the most important requirements for a program of this sort is, naturally, a thorough "knowledge" of its program domain. The consultant must know the structure of each of the Hermes objects, and the syntax and effects of each of the Hermes commands. Thus, at the very least, a representation is required that would allow us to encode this type of knowledge about Hermes in a machine-usable form.

A semantic net would be an apparently good candidate for this representation task. The touted strong point of networks is their

associative connectivity, and this is what is needed to express the relationships between commands, and between commands and objects. One could easily conceive of a hierarchy where the very similar PRINT and TRANSCRIBE commands could share a set of common properties, and they together with the LIST and SURVEY commands would inherit a set of still more general properties, etc. Further, nets are usually used to represent action-based domains, so that they stand a good chance of success at representing the effects of various commands (i.e., how the program execution proceeds).

Yet semantic networks as they have been commonly conceived still fall far short of the required expressive capacity when this domain is looked at in more detail. First, Hermes has several kinds of objects with complex internal structures -- and no techniques exist for representing the internal structures of things (as I have mentioned, nets have not been used to represent nominals, particularly in terms of their internal structures). We must provide a technique for the representation of structured objects. Second, there are complex interrelationships between Hermes entities that are not expressible when conceptual relations and concept-structuring relations are all uniformly links (for example, consider the representation of the statement, "the REPLY command takes the contents of the SUBJECT: field of the message being replied to, and, after concatenating that value with 'Re: ', makes that the contents of the SUBJECT: field of the outgoing DRAFT, except when the original SUBJECT: already begins with 'Re: ', . . ."). Third, the effects of commands must be stated in terms of potential values for arguments (i.e., descriptions) -- as we shall see in Chapter 7, power to achieve such description comes only from a change in basic approach to semantic nets. The definition, in advance, of many particular instances through a general description of legal potential fillers is an important way of deriving new concepts from old (existing) ones. Fourth, as the environment changes very often, the intelligent assistant must be able to track changes in its conception of the world, and constantly

establish connections between the particular objects that it knows about and the definitions of the corresponding object types. Thus a highly structured and very general individuation mechanism is required (see Section 4.3.3). As we shall see, this, too, depends on the approach one takes toward his representation. Finally, while it is necessary to have the intelligent agent know how Hermes works, the user who does not know the system will most likely classify the things that he wants to do in different terms than those in which the program is implemented. Thus the assistant must have the power to understand a user's functional conception of the Hermes world as well as the factual implementation details. This may entail more than one network; yet no precedents exist for meaningfully tying together multiple knowledge bases representing different conceptions of the same domain.

As with the document consultant, I will bear in mind in the next several chapters the requirements for a representation of this kind of knowledge. I shall point out how a sound foundational approach will alleviate some of the difficulties mentioned above, and will pave the way for a reasonable representation of knowledge about a computer program. I shall also illustrate how such a realistic, non-trivial domain can test a representation severely, and how the use of such a domain can unearth important fundamental problems with a formalism like the semantic network.

Chapter 4. What's in a Concept -- A New Foundation for Semantic Nets

As I mentioned in Chapter 1, my main intent in this report is to present a "Structured Inheritance Network" formalism for representing the kinds of conceptual knowledge needed to assimilate bibliography annotations or to assist a user learning about a message-processing program. In this and the next chapter, I present in detail the SI-Net formalism, and a semantic network-like notation for it. Section 4.1 introduces in a reference summary the entire scheme; however, it will not provide detailed justification for the particular links used, or for the particular "level" of knowledge represented by these links. The motivation for these will become clear only after we make an in-depth study of the foundational problems of the more traditional homogeneous nets in Section 4.2. The final section of this chapter will then review the formalism in light of the inadequacies thus exposed, providing a detailed account of the basic links that I proposed in 4.1, and showing how they can be used to avoid the shortcomings of the older notations. A similar discussion of links dealing with structure is subsequently presented in Chapter 5.

4.1. SI-Net notation

The particular notation that I will use in this report, like that of traditional semantic nets in general, is composed of nodes and links*.

* It should be borne in mind that the particular notation is expressive of the underlying content of the formalism, but it is not the formalism itself. The critical elements of the SI-Net idea are "concepts", "dattrs", and "structural conditions" (see below), and not the nodes and

The major difference between Structured Inheritance Networks and older types is the constrained repertoire of link types and the particular relationships that they represent. The only thing that the user defines using the this scheme is the set of nodes in the network. His nodes are of course tied together by instances of SI-Net primitive link types, but he cannot create new link types, nor can he construct arbitrary groupings of links at nodes. This is because the nodes are typed, and each node type has a fixed syntax for links that can emerge from it. This guarantees consistent interpretation by network processing routines, and gives us a criterion for conceptual well-formedness.

The nodes represent the places where the "knowledge" is concentrated -- in SI-Net notation there are concept nodes which represent predicates (and functions) for objects and actions, role description and role instance nodes which describe items that stand in important relationships to the concepts, structural condition nodes which express these relationships explicitly, and structural reference nodes which allow structured access to internal parts of complex descriptions*.

The central elements of SI-Nets are "concepts"; these represent the objects, the actions, and the relationships of the domain. A concept is considered to be a set of role/filler ("dattr") descriptions and a

links with which we might implement such epistemological abstractions. The reason a network notation is used (beyond its historical tie) is that it provides an explicit place for each possible kind of connection between two entities, and forces us to account for every epistemological relationship implied by the concept-dattr-structural condition paradigm. If I occasionally speak in this report of the node and link types as if they were synonymous with the underlying abstractions that they stand for, it is an imprecision for which I apologize.

* In the figures to follow, these node types will be differentiated by shape. Concept nodes are pictured as ellipses, role nodes as small squares, logical and quantificational nodes as diamonds, and special "parameterized" versions of each of these will have double borders. The structural reference nodes will be depicted as small squares, as they will be seen to be closely related to role nodes.

structuring gestalt which expresses the relationship between potential fillers of the role descriptions. I say "potential" here, because a concept is a template-like description that can apply to many particular objects in the domain; it is an abstraction of the common features of a group of entities that are perceived to be similar in some way. Therefore, a concept is a schematic description of a set of roles (including their potential fillers), and the way that fillers of those roles in particular cases will interact.

Potential fillers of functional roles are described in sets, each element of which is expected to play the same functional role within the structured object. The complex structure which describes the set of fillers and the functional role is called a "dattr" (for "description of an attributive part"). A dattr is a structure that allows us to speak of a set of fillers of a functional role in a given context, whereas "role" just refers to a filler's function.

Notationally, there are two types of links that tie a concept node to nodes representing the concept's internal structure. A dattr description is indicated by a link called "DATTRS" from the concept node to a node representing the description. The structural relationship (called the "structural condition", or "S/C")* is indicated by a "STRUCTURE" link from the concept to a structure which ties the dattr descriptions together in the appropriate way. As should be clear from this orientation, concepts are the representations of structured objects, with the dattrs describing the "parts" of objects (although not just physical parts) and the structural condition indicating how the

* In this report, I will collectively group all structuring relationships into a single structural condition. It does, however, seem useful to consider a segmented S/C in which various sets of relationships are grouped according to their intent or behavior with respect to interpreter processes (recognition, inference, etc.). Individual subparts of the S/C could then be assigned their own criteriality measures, could be inherited and altered separately, etc.

parts are put together.

For describing a set of potential parts of a structured object, a role description node is used. There are "internal" pieces of a such a node which represent the conjunction of the following information about a functional role to be played and potential players of that role:

- 1) a class of entities that are to be legally acceptable as fillers of the role. This is indicated by a "VALUE/RESTRICTION" (V/R) pointer to a concept node. Concept nodes implicitly capture sets of entities and the destination of the VALUE/RESTRICTION link delineates such a set. Any entity that can be described as a member of that set can fill the designated role.
- 2) the number of entities that are expected or required to fill the role in an instance. A "NUMBER" link points to a predicate that must be true of the number of role fillers in any particular instance of the concept.
- 3) the criteriality of the role to the concept as a whole. Some parts must be there to consider an entity an instance of the concept. Others are optional to the overall description; still others may be a consequence of the way that the object is structured, and are not independent entities. The notation allows a link called "MODALITY" to one of the values NECESSARY, OPTIONAL, or DERIVED.
- 4) the name and definition source of the the functional role. The notation requires a "ROLE" link to indicate the particular role that the part plays (e.g., AGENT, LINTEL, FUEL, etc.). The ROLE link points to another role description node of a more general concept. In the event that the current role is not defined elsewhere, a "ROLENAME" link will point to a string, to be considered as the name of the role. In that case, the role's definition is completely embodied in the structural condition (for details on this, see Section 5.1).

Fig. 4.1 illustrates a simple concept, which has two dattrs -- one specifying an open-ended number of potential role fillers, the other a single CARDINALITY*.

* In this and following figures, these conventions apply: 1) the label in a concept node represents the node's print-name. A label that is enclosed in parentheses indicates the derivation of the concept from other concepts in the network (Woods calls this the "EGO", but see Chapter 5 for an alternative explanation). 2) Roles enclosed in

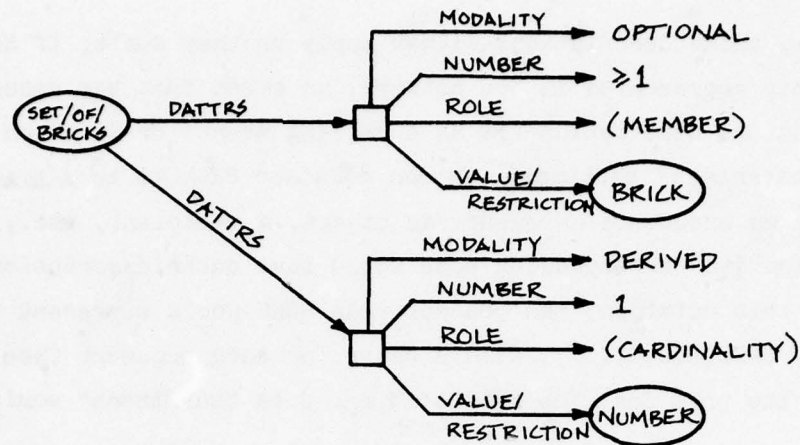


Figure 4.1. A simple concept.

Dattrs, as I have mentioned, can describe attributes of an entity other than its parts. For example, if we wish to define the concept of an ARCH*, we might like to include a role for its vertical clearance, since in many applications, the height of things that can fit through an arch is critical. The clearance, however, is a product of how the arch is built, and is not something that goes into the making of the arch itself. As such, we shall consider the vertical clearance a derived dattr of the concept.

There are two special types of derived dattr that we might consider. Besides knowledge of particular facts and entities in the world, our memories include predicative descriptions of classes of entities, which "apply" to objects in the world -- these are our generic concepts. In

parentheses indicate role nodes not shown. 3) If no MODALITY link is specified, the corresponding dattr is assumed to be necessary; 4) if no NUMBER link is specified, the default is 1; and 5) values not enclosed in ellipses or parentheses (except for modalities and number predicates) are intended to be only suggestive, and should not be taken literally.

* In examples throughout this paper, I will make use of Pat Winston's [1970] notion of an ARCH -- two rectangular bricks supporting a third brick. This simple "blocks world" concept will help to illustrate some of the structural features of the network notation.

many cases, these descriptions either apply or they don't; if SELL is a relationship represented in our network, an event that has occurred either fits the description (it is a selling event) or it doesn't (it is not and instance of selling). We can consider SELL to be a predicate that takes as arguments an agent, an object, a recipient, etc., and in our notation its corresponding node would have dattr descriptions for each. In this notation, the concept node SELL would represent the predicate, SELL(A,O,R,...), with a dattr for each argument (see Fig. 4.2), and the node for "The Nets sold Dr. J to the Sixers" would

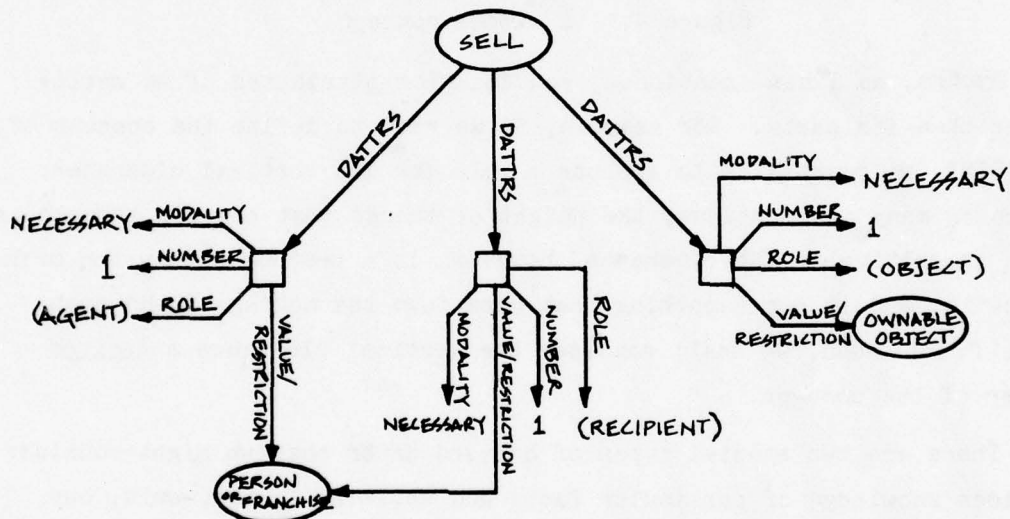


Figure 4.2. A predicative concept.

represent the proposition, which is a filled-in version of the predicate (I will illustrate the way to represent propositions in Section 4.1.2). This tells us that a SELL relationship exists between the parties named. In this case, SELL has no derived dattr.

In other cases, however, we do not expect a concept to simply "apply" or not, but upon its application we want it to return us a value. DISTANCE, for example, is such a function. The distance between two points is a number with some units, some measure of how far apart the two points are. Thus the concept for distance must have a dattr

description for the value returned, which is to be derived from the two points. Thus, while the DISTANCE function takes only two arguments (i.e., its NECESSARY dattrrs), it has a total of three dattrrs (see Fig. 4.3 -- node R in the figure indicates the value that is the distance).

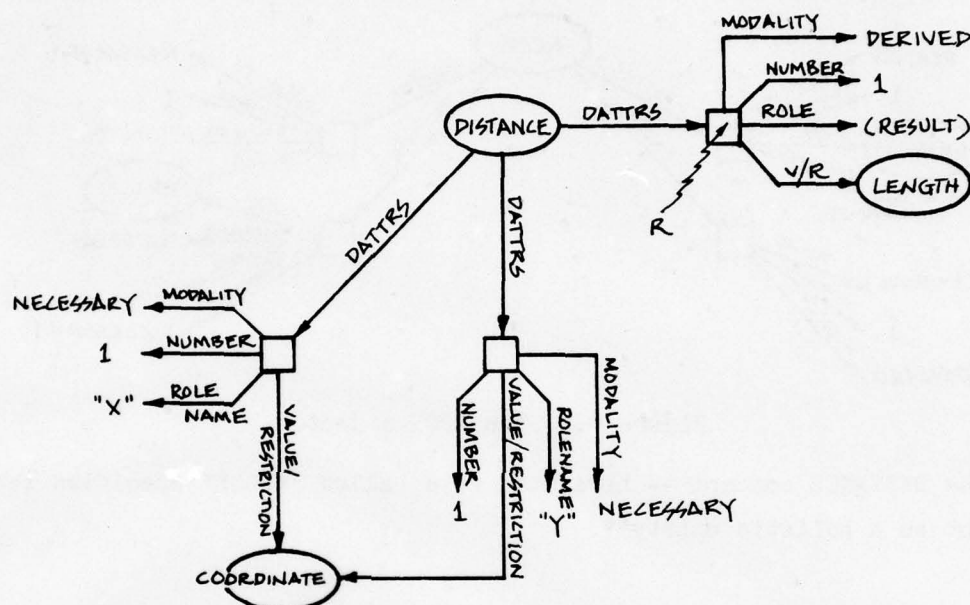


Figure 4.3. The function DISTANCE(x,y).

Finally, given an object like the ARCH, we need to account for a slightly different aspect of the concept. We might imagine a predicate, ARCH(x,y,z), which, when applied to three bricks, tells us whether the "arch" relationship can be considered to hold among those three bricks. We can even imagine an "arch function", which returns as its value the arch that exists there. However, there is still another type of predicate that we can consider, the one embodied in the question, "Is that thing an arch?" -- that is, a predicate which takes a single argument, which is an "object", and which we determine to be an arch or not. This notion of "objecthood" is still a bit mysterious, and is a result of our own perception rather than of the structure of the world. Yet it is one of the most important "myths" that we have for dealing

with our environment, and it affects most of our thoughts of the universe*. Thus, the notation should probably provide a description which is to be thought of as the thing which is the arch. Fig. 4.4 illustrates how we might conceptualize ARCH in a manner similar

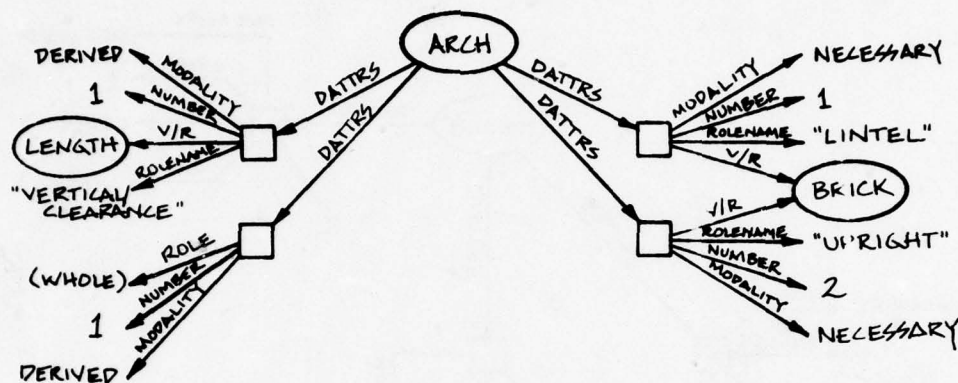


Figure 4.4. The ARCH object.

to the DISTANCE concept -- here, the role called "WHOLE" specifies the object as a holistic entity**.

* "But in point of epistemological footing the physical objects and the gods differ only in degree and not in kind. Both sorts of entities enter our conception only as cultural posits. The myth of physical objects is epistemologically superior to most in that it has proved more efficacious than other myths as a device for working a manageable structure into the flux of experience." [Quine 1953, p. 44]

** We might contemplate the use of the WHOLE dattr to indicate relationships in which the entity as a whole participates (see Fig. 5.1, for example). It is DERIVED in the sense that the whole is not an argument, and is a result of putting all of the parts together in the way specified by the structural condition (see below). However, since it is the "whole" thing, that is, the sum total of all of the dattrs and their interrelationships, it seems imprecise to think of it as a "dattr". I will not pursue this further except to point out that the same discomfort is evident with languages like KRL, which have "slots" called "SELF" (see Chapter 8).

4.1.1. Structural conditions

The structural condition (S/C) of a concept is an explicit description of how its role fillers go together. This description is expressed in terms of other concepts that exist elsewhere in the network, and thereby captures the way that we describe what we know in terms of other concepts that we are familiar with. If the concept being defined is a predicate, then its structural condition describes the relationship that must hold between the arguments for the predicate to apply; if a function, the structure determines how to compose the arguments into the result to be returned as the value of the function; and if an object, it describes how the parts go together to make it the kind of object it is (the "gestalt"). In all cases, the structural condition determines how to derive any DERIVED dattr from the arguments.

As I have said, structural conditions express relationships by utilizing other relationships already extant in the network. By pointing to role description nodes of their enclosing concepts, structural conditions help define the functional role parts of the dattrs represented by those nodes. We can begin to express this kind of definition as in Fig. 4.5, with the STRUCTURE link in general linking the concept node being defined directly to special tokens of concepts defined elsewhere. In this case, ARCH is the concept being defined, and the structural condition node labeled "(SUPPORT)" stands for a special kind of concept called a "Parametric Individual", which is intended to represent the particular version of SUPPORT that is relevant to ("parameterized by") ARCHes*. In the simple structural condition of

* I discuss the precise form for nodes for individuals in the next section, but for now it will suffice to interpret the "DINSTS" link in the figure as a pointer to an attribute/value pair which designates how a role is filled. The DINSTS link corresponds to the DATTRS link, and specifies the instantiation of a dattr by a particular value in a particular concept. Also in Fig. 4.5, the ParaIndividuator of SUPPORT does not have its own name; it is labeled "(SUPPORT)" to indicate its source, and in later figures I will often omit the full derivation of

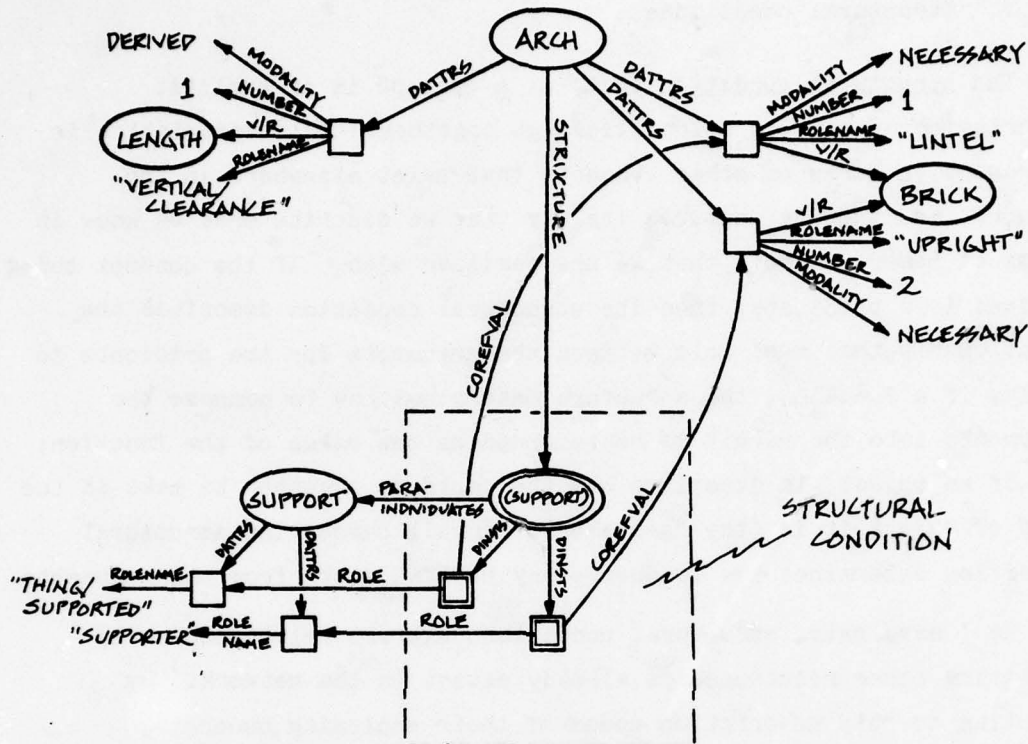


Figure 4.5. Incomplete structure for an ARCH.

Fig. 4.5*, the two links from the (SUPPORT) node's role nodes to the role description nodes of the enclosing ARCH concept indicate how the arguments to ARCH must be related. A "COREFVAL" link to a role description node is interpreted as specifying the fillers of the role in particular cases of the concept. Thus, in any instance of ARCH, the particular LINTEL of that arch must be supported by the particular UPRIGHTs of the same arch. The structural condition and role

such a ParaIndividual and rely on the parenthesized label as an abbreviation for the more detailed structure.

* The dotted lines delineate the extent of the structural condition, and are for illustrative purposes only. These lines make the S/C look like a partition [Hendrix 1975a, b, etc.] -- this is not particularly the intent, although S/C's could be implemented that way.

descriptions describe a general pattern to be fit by each instance of the concept.

Notice that the elementary structure of the figure above does not yet express how the VERTICAL/CLEARANCE can be derived from the structure of the ARCH. In addition, notice that since there are two UPRIGHTs in each ARCH, the SUPPORT token in the structural condition implicitly expresses a quantification over the fillers of the UPRIGHT role. Thus, to be more precise, we need to augment the structural condition with logical connectives and quantifiers. To this end, there is a set of special concepts that express things like conjunction and disjunction of several predicates, negation, equivalence, and universal quantification. These concepts are structured and used just like ordinary concepts, except for the fact that their own structural conditions will be considered primitive, and not represented in this formalism (because their definitions would be circular).

These operators are used in the obvious way*, and Fig. 4.6 illustrates the required augmentation to the structural condition of ARCH. The center conjunct of the (AND) node** expresses the same SUPPORT relation as above (Fig. 4.5), this time with the quantification explicitly broken out: the (EVERY) node shows two dattr's, "x" and "P"; x specifies the class of entities over which the quantification is to

* See Schubert [1976] for a similar but more complete scheme which is analyzed in detail.

** The three conjuncts here are indicated by DINSTS links, as are the role fillers for all of the logical operator nodes. While the fillers of these roles are not really individuals, they exhibit the same inheritance characteristics as instantiated dattr's generally do. That is, their use in the particular structural condition prohibits their further modification or instantiation, so they in this sense act like individuator's. It is for this reason that I have used the DINSTS link. These pattern-like tokens are called "Parametric Individuals" because they are, in a sense, parameterized by the concepts in which they appear.

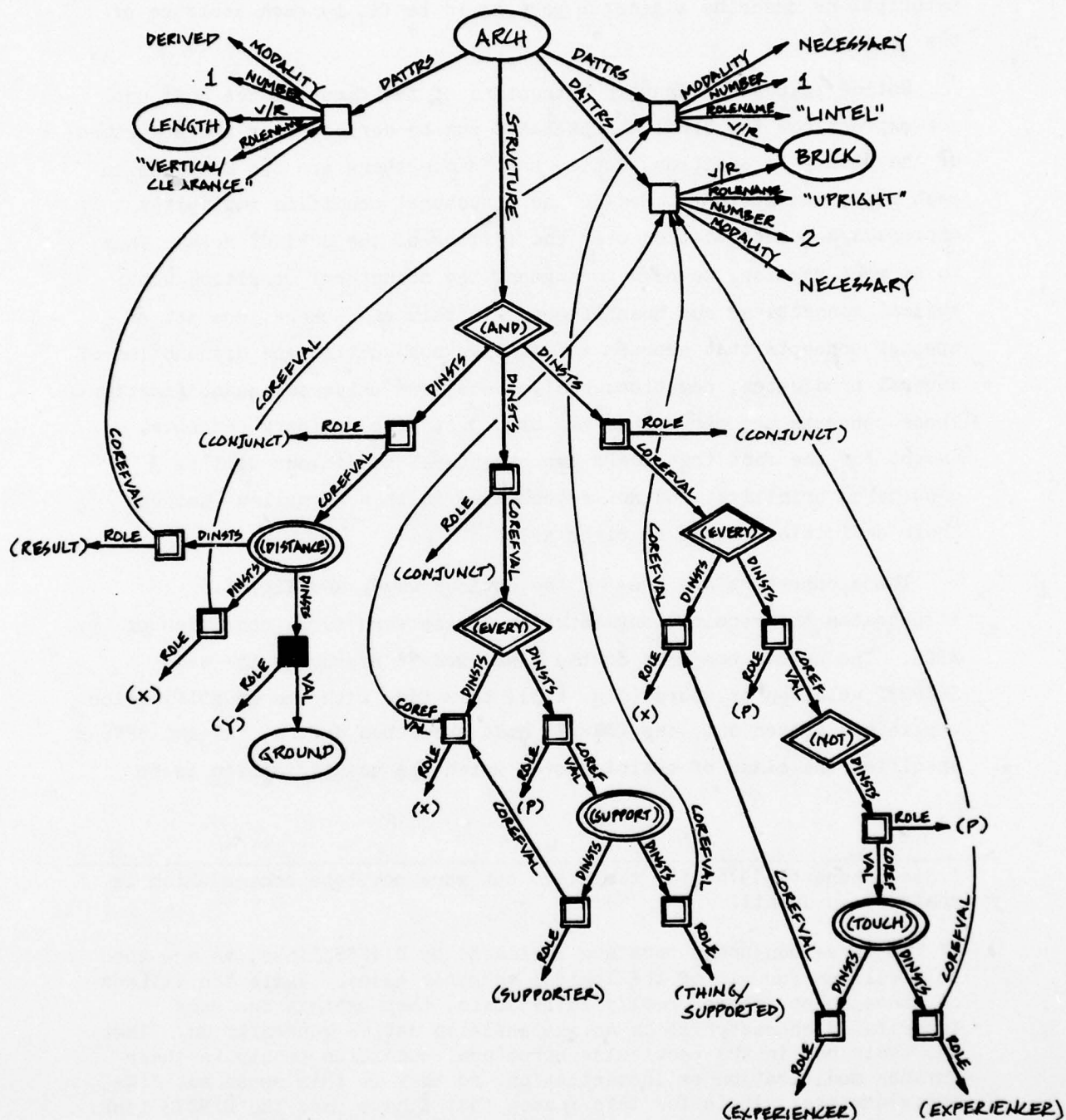


Figure 4.6. Detailed structure of an ARCH.

Section 4.1.1
Structural conditions

range, and P specifies the predicate to be applied to each x (notice how one role node of (SUPPORT) is linked to the "x" node). I shall return to this in a moment.

The rightmost CONJUNCT in Fig. 4.6 expresses the fact that no upright must touch another upright. The leftmost CONJUNCT uses the DISTANCE function to express where a VERTICAL/CLEARANCE comes from -- it is the RESULT of the distance function applied to the LINTEL of the arch and to the constant, GROUND.

The special concepts do not cover the entire spectrum of quantification (as I mentioned, see Schubert [1976] for a more complete set), but are intended only as a suggestion on how to build structural conditions (as we shall see in Chapter 7, however, they are sufficient to cover a broad range of phenomena). Schematic definitions of the five concepts that I shall use are shown in Fig. 4.7. The EVERY quantifier has three dattr, one to specify a class from which the variable is to be taken (x), one to indicate a restrictive predicate to be applied to each value of x (R), and one to specify the predicate that is to be applied to each class member that passes the restrictive predicate (P). This reflects Woods' [1968] "FOR" notation,

FOR EVERY x / CLASS : R(x) ; P(x)

which reads "for every x in CLASS such that R(x), P(x)".

One final point to be made about concept structure is the

* Existential dependence on a universally quantified variable can be expressed in the notation if the "Skolem function" technique of formal logic is used (see [Woods 1975a, p. 76] for an explanation of this technique). With this technique, each existentially quantified variable is replaced by a functional designator whose arguments are the universally quantified variables on which the existentially quantified one depends. We can mimic this structure with our notation. Consider the statement, "for every x there exists a y such that P(y)." If y is replaced by some function, f, of x, then every time y appears in the quantified expression, P, we would have a pointer to the RESULT dattr of the node representing f(x). See Fig. 5.6 for an example.

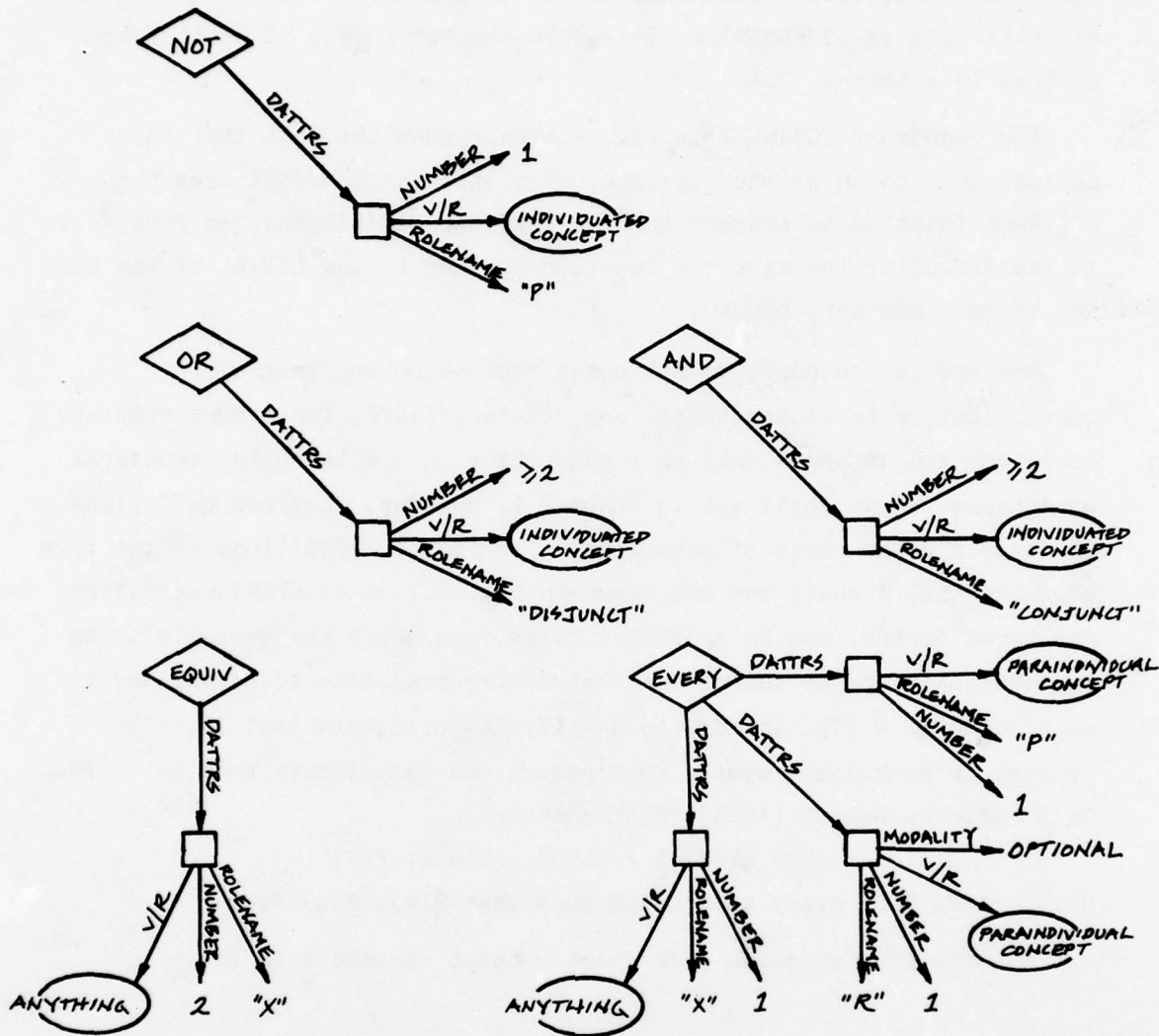


Figure 4.7. Logical operators.

possibility of more complex accesses from the structural condition to the role descriptions. In Figs. 4.5 and 4.6, I have indicated the potential fillers of a role of the ARCH concept by simply pointing directly to the appropriate role description node. But consider a case where, say, we wanted to point not to LINTel, but to one of the subparts of the LINTel brick. If a brick were considered a structured object, with a set of PLANEs (its faces) as its dattr's, we might want to point

to the BOTTOM PLANE to make our vertical clearance derivation more precise. At first, we might simply point to that aspect of BRICK, as in Fig. 4.8. Unfortunately, we couldn't tell, then, if the distance were

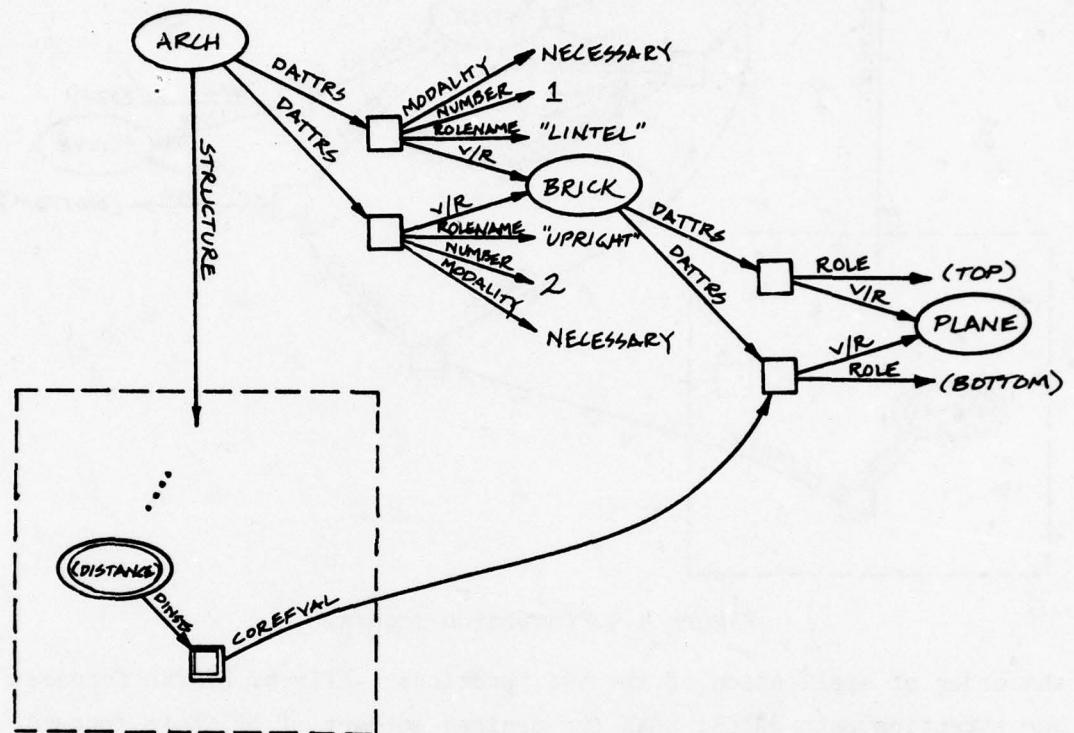


Figure 4.8. Access ambiguity.

to be taken from the bottom of the LINTEL or from one of the UPRIGHTs, since both role descriptions point in the same way to BRICK!

If we consider a role description node to embody a dattr access or focusing function that focuses on the particular aspect of the concept, what we need is a composite dattr function. Such a function is provided in the notation by a special use of the role description node, as illustrated in Fig. 4.9*. The two links, FOCUS and SUBFOCUS, indicate

* A role description node is used to enable recursive functional composition. Thus a FOCUS or SUBFOCUS link can point to a regular role description node, or to another composite dattr function.

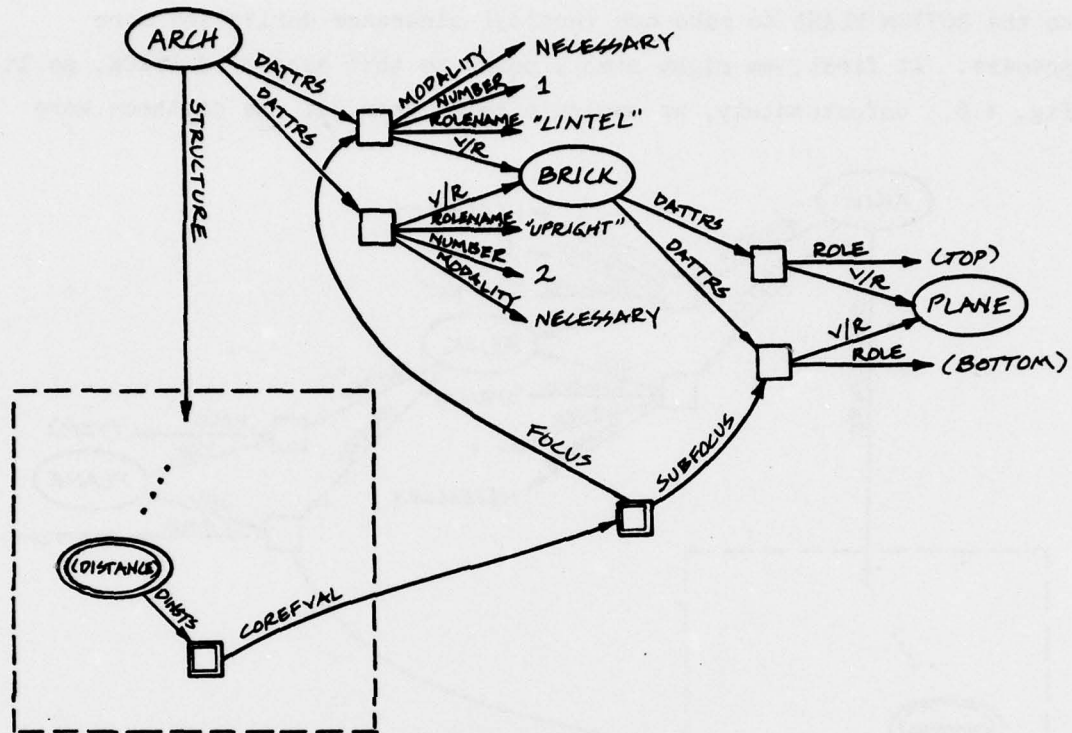


Figure 4.9. Composite access.

the order of application of the two functions -- first, LINTEL focuses our attention onto BRICK; then the desired subpart of BRICK is focused on by BOTTOM. The composite function thus formed has an implicit argument, namely the concept from whose structural condition it is accessed.

The above links form the nucleus of the SI-Net representation of structured objects. They provide the basis for a representation of the "internal structure" of concepts.

4.1.2. Inter-concept relations

Concepts also participate in relations with other concepts. Our main concern with this type of relationship is the expression of the derivation of new concepts from already existing ones. Definitional relations between concepts are indicated by "INDIVIDUATES" and "DSUPERC" links, which indicate the inheritance of various pieces of "super"-concepts to "individuators" and "subconcepts", respectively. One of the principal tenets of Structured Inheritance Networks is that such inheritance ranges over a set of structurally determined elements (i.e., the dattr and S/C of the superconcept), and that no individual "property" is inherited by itself, independent of the conceptual complex as a whole. This is what makes the network a structured inheritance mechanism. Therefore, the "link" that channels the inheritance from a concept to its descendant is more like a cable. While our figures will suggest the independence of the inheritability of dattr, bear in mind that no dattr can be inherited without this "cable" being present.

An indivuator of a generic concept is a concept with all role descriptions filled by particular values; in addition, an indivuator purports to represent a single entity in the domain (the domain entity is considered to be an "instance" of the generic concept). Individuators are always defined relative to generic concepts* and this relationship is the import of the INDIVIDUATES link. When a particular entity in this way satisfies the description embodied in its defining superconcept, each role filler description in the indivuator must be mapped onto the generic role description that it satisfies. This is indicated in the older nets by simple "attribute/value pairs"; but in SI-Nets one conceives of an "attribute" as a complex entity (the

* That is, a concept is not an indivuator in and of itself, but only by virtue of its individuating some generic concept. Thus, the term "indivuator" is used analogously to the term "subconcept".

The diagram illustrates the relationship between a 'GENERAL CONCEPT' (ARCH) and an 'INSTANCE' (ARCH59). The diagram shows a hierarchy where ARCH is the parent concept, and ARCH59 is a specific instance. ARCH has three roles: 'VERTICAL/CLEARANCE', 'LINTEL', and 'UPRIGHT'. ARCH59 has three roles: '3 FEET', 'B', and 'C'. The diagram also shows a 3D perspective of a structure with sections A, B, and C, and a height of 3 feet.

Figure 4.10. An individuator and its defining concept.

* This is one of the most important features of SI-Nets. In almost all other network representation languages, role information is supposedly carried by atomic link names. In the view being developed here, the functional role/filler complexes can be considered to be complex "names" which can be pointed to, and which have internal structure. In addition, since these pointers are explicit, more than one role description could have the same functional role name (string), yet each could refer to distinct aspects of the concept. As we shall see, this is not possible with the standard attribute/value conception.

INDIVIDUATES link joins the two concepts together, and for each required role filler of ARCH, ARCH59 has a corresponding "DINSTS" link to a role instance node (a filled-in square; note the "extra" role instance node in this figure -- recall that Fig. 4.6 specified that an ARCH requires two fillers for the UPRIGHT role). The ROLE links are explicit pointers to the "attributes", and the VAL links indicate the role fillers for the ARCH59 case of ARCH. The INDIVIDUATES link means that its source node represents a unique entity in the domain being modelled, and that that entity satisfies the predicate implicit in the defining concept. In that case, no structural condition need be indicated for the individuator, because by virtue of its fitting the definition, that structure is inherited implicitly from the parent. Each ParaIndividual in the structural condition of the parent would become a real individuated concept for the individuator.

There are other ways besides individuation to relate two concepts. The DSUPER link represents a definitional connection between two nodes that allows the subconcept to act itself as a predicative concept (and thus have individuators and further subconcepts). Therefore, rather than satisfying the requirements of the parent, the subconcept inherits them from the superconcept and can further modify them. A subconcept can perform three different operations on the inherited role descriptions:

- 1) restriction -- the "DMODS" link points to a role description node which specifies some further requirements for one of the roles; these are to be interpreted in conjunction with the requirements of the parent role node (indicated by a ROLE link from the role modification node to the parent role node). Thus, a further restriction on, say, a VALUE/RESTRICTION can be indicated (see Fig. 5.8, for example);
- 2) role differentiation -- a role of the parent concept may have (several) subroles that are to be explicitly distinguished in the subconcept. The "DIFFS" link points to a role description node which has both ROLE and ROLENAME specified. The former indicates the parent role that is to be subcategorized, the latter the name of the more specific subrole. For example, in the definition of a CAR, we may wish to take the WHEELS role of the parent concept,

VEHICLE, and break it into FRONT/WHEELS and REAR/WHEELS. Fig. 4.11 illustrates how to express this. The meanings of FRONT and REAR

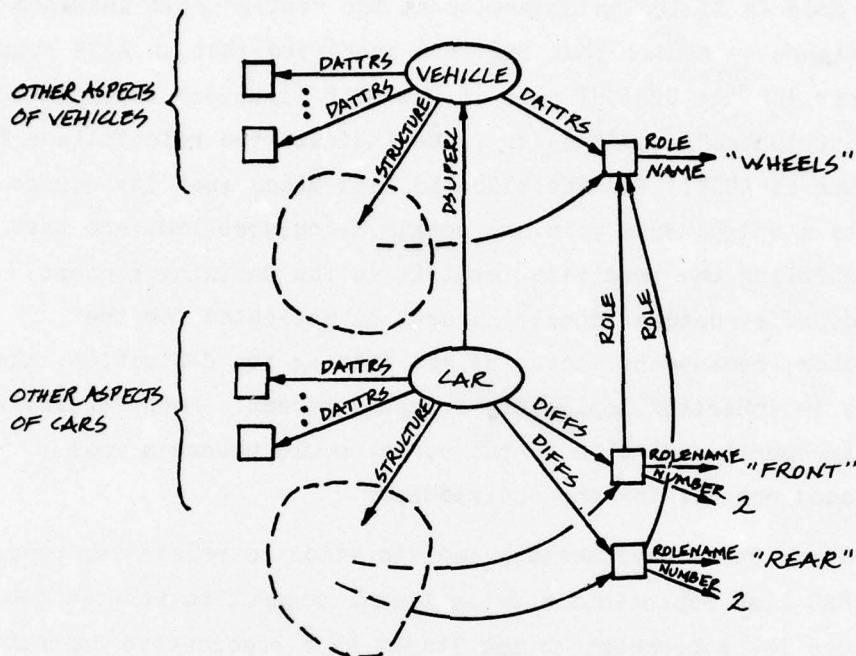


Figure 4.11. Differentiated roles.

are embodied in the structural condition of CAR, while the meaning of WHEELS is embedded in that of VEHICLE;

- 3) particularization -- rather than alter the description of the potential fillers of a role, we may wish to define a concept by specifying the value of one but not all roles of a higher concept (e.g., a REDHEAD is a person whose hair color is RED). This operation is the same as used in individuators and thus I use the DINSTS link in the identical way to indicate the particular value for the role (see Fig. 5.10). No node below the subconcept can point with a ROLE link to the role instance node thus indicated -- the value of the filler is itself inherited directly, and the particularized role is considered to be filled for all subconcepts below the concept containing it.

A third way of relating two concepts is the use of one within another as a "parameterized" pattern for generating the structure of instances. A "Parametric Individual" (ParaIndividual) is derived from

its superconcept in much the same way as a normal individuator is, except that 1) it must appear within the structural condition of some concept, and 2) its role nodes can point in a special way to the role description nodes of the enclosing concept. The role nodes in the ParaIndividual that are linked to role nodes of the enclosing concept by "COREFVAL" links specify that in any individuator of that enclosing concept, the role filler specified for the individuator is considered to be "coreferential" with the role filler of the ParaIndividual. In other words, for each individuator, there is an implicit version of the ParaIndividual that corresponds to that individuator alone, and whose roles are filled as specified by the COREFVAL links. The paraindividuation relationship is expressed in the notation by a "PARAINDIVIDUATES" link (cable).

Still a fourth way that a concept can be derived from another is by analogy. In such a description by comparison*, most aspects of the two concepts are assumed to be similar, with only the ones that are different being explicitly pointed out ("x is like y, except for its z"). SI-Net notation includes a link called "DBROTHERC" to allow such an analogy to be encoded directly in the network. It works like DSUPERC, but instead of pointing to a "parent" concept, it points to a "brother" concept, all of whose role descriptions and role instances are to be inherited intact, except for those explicitly pointed to by ROLE links. Instead of modifying or particularizing the roles of the brother node pointed to with ROLE links, however, the new brother applies its DMODS, DIFFS, and DINSTS to the parent of its brother. Thus the ROLE links are, in a sense, transitive. The DBROTHERC link is like an abbreviation for a DSUPERC link from a concept node that looks exactly like the indicated brother (except for the changes, of course), except that it also provides an explicit correspondence between the dattrs of

* This is relevant to, but not the same as, the KRL "perspective". See Chapter 8 for details.

the two similar concepts. The critical difference between a DBROTHERC and a mere copy of the brother concept is the fact that any changes to the brother are inherited by the new concept (they remain "in sync").

4.1.3. Relating nominal and verbal concepts

The final set of links is oriented toward deriving nominal concepts from verbal ones. As I mentioned at the outset, SI-Net concept nodes can represent equally well structured objects and actions. In Chapter 7, I will detail how the notation that I have presented can handle adequately the use of objects by specific actions. However, there is an important class of relationships between nouns and verbs that are definitional, and these relationships are represented by links in the notation itself. This is the set of transformations called "nominalizations" -- in Chapter 6, I discuss in depth the various kinds of nominalizations that are important to represent; here I only briefly mention the links that are proposed. The reader is encouraged to thumb through Figs. 6.3 to 6.11 for detailed illustrations of these links.

Nominalization links in SI-Net notation pass restrictions and role fillers, much the way that DSUPERC and INDIVIDUATES do (and are therefore also thought of as "cables"). They also indicate in which way a verbal concept or a particular event (an instance of a verbal concept) should be talked about as a thing unto itself (i.e., as a structured object). The concept of a particular event's having occurred can be spoken of as a fact; this interpretation is indicated by a link called "DFACTIVE". An event can produce a concrete object as its product, whose interpretation depends on the event itself (e.g., a drawing, laughter, marriage, etc.). I will call this nominal a substantive, or result, nominal, and indicate it with a "DRESULT" link.

The activity itself might be our concern, either as a process (e.g., destroying), or as a completed action (e.g., destruction). These two

interpretations are indicated by "DACTIVITY/PROCESS" and "DACTIVITY/COMPL-ACTION" links, respectively. "DGEN" (generic) indicates an important subcategorization of activity nominals. It defines a generic singular entity which represents a kind of event in general (e.g., the swimming of the English Channel); an individuator of the generic form can represent a single hypothetical event, which may or may not have transpired, and whose referent is unknown (e.g., an orbiting of Mars). The final kind of nominalization indicated by a link is that normally reflected in words like "maker" and "graduate" -- a participant in an action whose name is derived from the action itself. These nominals are derived not from the verbal concept, but from its roles. I will use a "DROLE" pointer to a role description node to indicate this kind of derived nominal concept (AGENT in the case of "maker", OBJECT in "graduate").

This completes the summary presentation of the basic notation. Chapters 6 and 7 provide detailed examples of its use and power to express the structure underlying two different domains. Those chapters will illustrate how SI-Net notation handles the issues of structured objects, idiosyncratic definitions, uses of nominal concepts by verbal ones, and some paraphrase retrieval operations.

I now turn to the motivation for this new notation. In the remainder of this chapter I discuss the justification for a new approach and for a particular set of links and nodes, by analyzing in detail what "concepts" are. I begin this analysis by determining why the older type of network is inadequate, and consequently motivating an "epistemological" approach to the foundations of networks.

AD-A056 524

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MASS
A STRUCTURAL PARADIGM FOR REPRESENTING KNOWLEDGE.(U)
MAY 78 R J BRACHMAN
BBN-3605

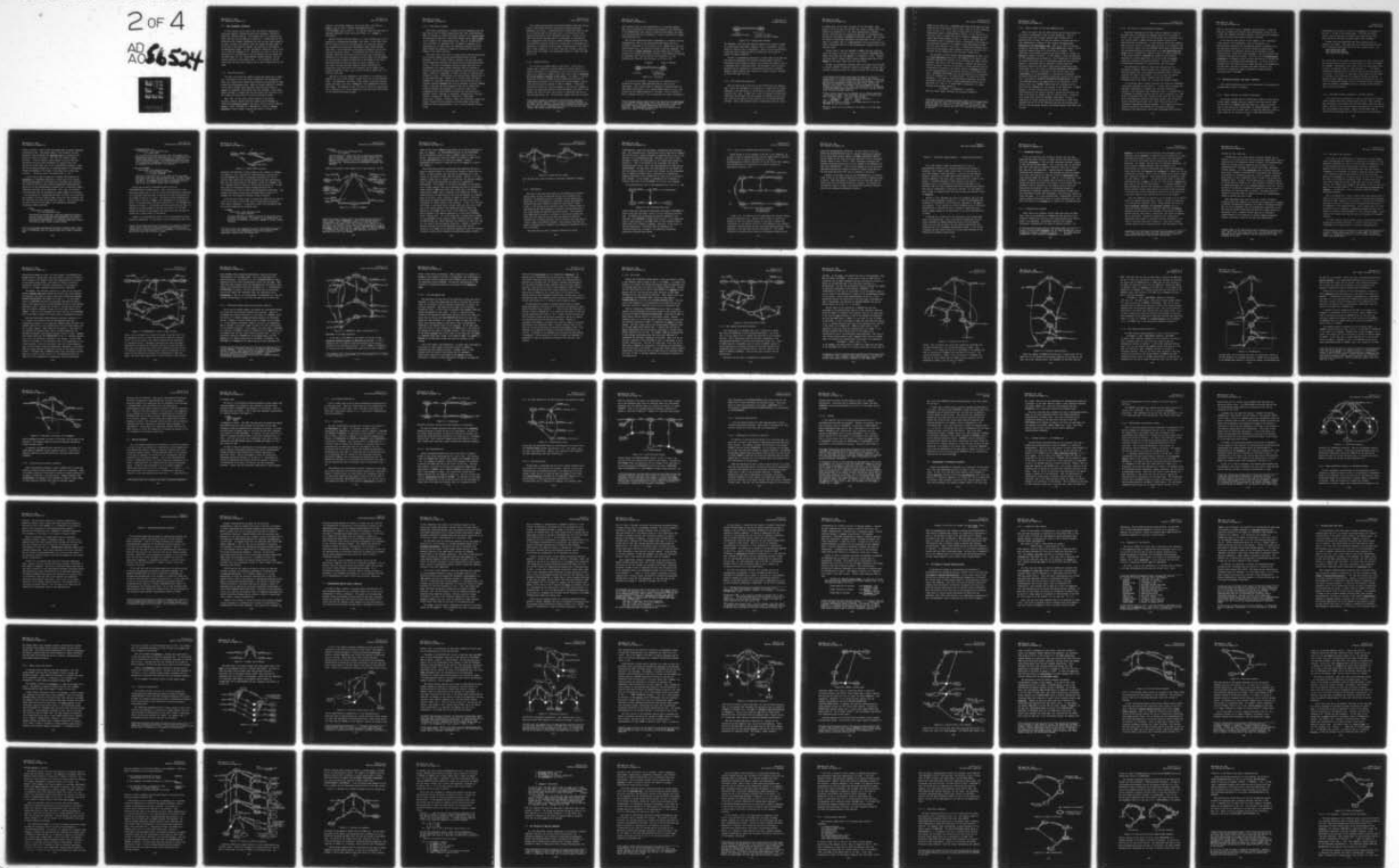
F/G 5/6

UNCLASSIFIED

N00014-77-C-0371
NL

2 OF 4

AD
A056524



4.2. Some fundamental confusions

In this section, we will look at how the primitive operations of traditional semantic networks are most often confused or obscured by high-level, uniform notations. Our approach will be to concentrate on nodes, and what they are expected to stand for -- network nodes are the places where the "things" represented by a net are most often claimed to be found. We shall see, however, that even though the nodes appear to be the seat of the network's content, the links end up carrying the representational weight. The only information "stored at" a node is the set of links that impinges on it. Therefore, the links in the net are responsible for representing not only the operations on the things that the nodes stand for (e.g., "associations" between concepts), but the internal structure of the nodes as well. I will propose to use the links solely for this latter purpose, and to leave "associations" to the nodes.

4.2.1. What are nodes for?

The basic idea behind the semantic network has always been a simple one. The objects of the world under consideration are represented by nodes, and "associations" between those objects are represented by links between the corresponding nodes. Nodes are usually labelled to indicate to the network designer their meanings; links also bear labels to suggest the conceptual relationships that they represent. Given such a general paradigm, it appears that one could tailor the relations in his net to make the nodes stand for virtually anything he wanted.

What, then, are the nodes normally expected to represent? Typically, semantic network nodes are places at which knowledge is stored about particular things in the world: there are usually nodes for objects ("John", "Telephone T1", "Message 16"), nodes for factual assertions ("John's height is greater than Mary's", "Brazil is a

Section 4.2.1
What are nodes for?

country", "the default message is the current one"), and nodes for events ("John hit Mary", "Message <[BBN-TENEXA]22-Jun-76 14:02:48.HERMES> sent at time T"). In addition, nodes are often used to represent groupings of these particular things, i.e., classes of individuals.

Besides linking individual members to a class, networks give us the capability of representing subclasses. The result, if the links in the net are viewed as arcs of a directed graph, is a tree-like structure with subclasses and individual class members linked "under" general class nodes. This hierarchical layout gives the semantic net its most prominent structural feature. Its advantage lies in the ability to represent assertions about many entities at one time; by linking some information to a node which represents an entire class, the net designer can avoid having to repeat an assertion for each member of the class. This application of assertions about the class in general to a particular individual has been generally called "inheritance of properties", since the node for the individual can be thought of as having the assertion (that it has a certain property) passed down to it from the class node.

Thus one of the most fundamental relationships to be expressed by a semantic net link is that between a member object and its corresponding class. This membership link comes in many guises in different semantic network notations, with "ISA", "MEMBER/OF", and "INSTANCE/OF" being the most common. The meanings of such links may be obvious to human readers of the notation, but it is not immediately clear what their implications are for network-processing programs.

4.2.2. From class to concept

Upon closer investigation, it appears that the membership link has been tacitly used to express significantly more than class membership. Nodes for classes are almost universally referred to as "concept nodes", the implication being that a node should somehow capture what it means to be a member of the corresponding class. That is, the generalized idea of an ARCH, or a WALKing action, or REDness is supposed to be described by one of these nodes, and individual instances of each of these concepts are expected to assume the characteristics known for members of the class in general.

This subtle shift from class to concept carries serious implications for the entire network notation which heretofore have not been addressed. If all we really wished to express in the net were subset and set membership relations, the obvious lattice representation would suffice. But since its designers implicitly expect *much more* to be representable in a single network formalism, they are required to produce an expressively adequate and logically consistent way to define "what it means to be something" in terms of a group of links that are attached to a node. Unfortunately, none exists, mainly because while it is well understood what a "class" is, it is never clear just what a "concept" is. Quillian's expectation that his nets would be able to uniformly represent anything that could be expressed in natural language [1969, p. 460] has led to the assumption that one can simply take virtually anything, and implement it in nodes and links; there consequently have existed semantic networks that have purported to represent "facts", "meanings of sentences", "propositions", "actions", "events", "properties", "wants", "tendencies", "assertions", "predicates", "objects", "classes", "sets", and "relations", among other things. "Concept" has thus appeared at various times to mean some indefinite kind of generalization of each of these different kinds of things.

Can we glean anything about the precise meaning of the term from the forms in which these so-called concepts have actually appeared? Apparently very little -- statements about concepts have tended to rely on our intuitive feel for the term, and so it is rare that we see a detailed discussion of their implemented structure. At best, we can infer from the way concepts seem to be interpreted in existing nets that, regardless of what they are, network designers believe that they can be defined as groups of features or properties, or occasionally as predicates. In addition, while we might occasionally get a fair idea of what a verbal concept is, it is never clear what to make of nodes for nominal ones (for example, the underlying structure of nodes like JOHN, RED, etc.)*.

4.2.3. Property notation

Let us look at "concept nodes" in more detail. Intuitively, a concept node is supposed to express somehow the general nature of a class of individuals. By expressing all and only the qualities that it takes to be counted as a member of the class, such a node is describing all of the potential instances of the concept. For example, a node for the general notion of a Hermes printing command would describe the common parts and features of all particular printing commands, and how those features were put together to make a "command". These might include the relation of a printing command to the notion of command in general, the specialized syntax of printing commands, the nature of the objects that they can print, their effect as outputting some text on some printing device, etc. Any particular command that were to exhibit

* The notable exception is Winston's [1970] structural paradigm, although he doesn't appear to use a uniform structuring mechanism for his nodes, and the structure confuses nodes of different varieties (see below). More recently, languages such as KRL and FRL have made some inroads into nominal concept understanding.

these features could be then characterized as a "printing command". This kind of description of potential class members is the abstraction of the commonalities from a group of particular individuals, and seems to be the prevailing way of thinking of a class of entities as a single representational unit.

How are these abstracted features implemented in a semantic network representation? Each of the features mentioned above would be a "property", consisting of an "attribute" (e.g., SYNTAX) and a value for that attribute (e.g., the particular syntax of printing commands). Now, in the case of a single individual, the most common type of network notation would reflect the property directly in the network by attaching an "attribute link" from the node for the individual to a node specifying the value for that attribute, as in Fig. 4.12*.

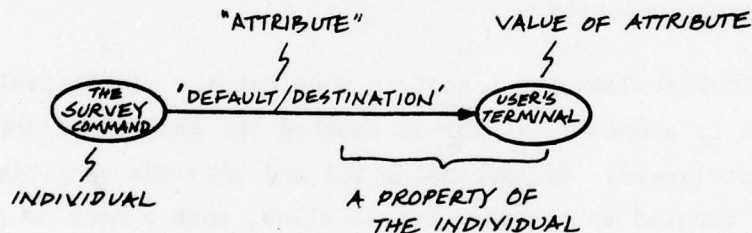


Figure 4.12. A property.

As we see in this figure, the properties of individuals can be expressed by attribute links emanating from the nodes for the individuals. Since concept nodes are supposed to represent groups of individuals collectively, it is a natural generalization to use a like notation for the common properties of the group. Such a notation is easy to generate (see Fig. 4.13), and promotes the overall uniformity of

* This treatment applies equally well to the definition of common parts of instances, and I shall assume in the discussion that follows that what holds true for attribute description also holds for part definition. The commonality between these two is the basis for the notion of a "dattr" -- see Section 4.3.

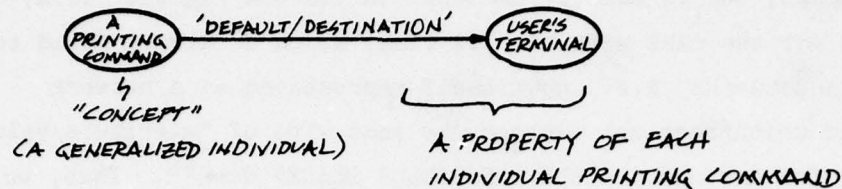


Figure 4.13. A generalized property.

the formalism. Note that this decision compels us to look at a concept as a "generalized individual" (commonly called a "prototype"), and no longer as a group or set of individuals. (This is one of the notational tricks that can be played by implicit conventions, and we must be careful to remain consistent.)

The move to representing properties at concept nodes in the same way that they are represented at so-called "instance nodes" (nodes for individuals) surely seems the logical way to extend the notation, and is a technique that is universally used. Unfortunately, there are subtle problems with the expression of these general qualities in standard semantic network notation that preclude our hastily trying to represent an entire data base this way.

4.2.4. The trouble with properties

Now, we finally get to the heart of the difficulties with semantic nets. First, the attachment of a property to a concept node is usually expressed in the same manner as the indication of the class membership relation (e.g., ISA). A named link is used to represent the particular attributive relationship that holds between the concept and some value (the link plus the value is often referred to as an "attribute/value" pairing). Such a link therefore names a relationship which can itself

be talked about, and is thus a "concept" in its own right*. This, however, is not the case with the ISA link, which is always found to be primitive in networks (i.e., not itself represented as a network entity), and which does not express the same kind of "attribute/value" pairing as, for instance, <TELEPHONE COLOR BLACK> does**. Thus, we have two kinds of relationship expressed by the same type of link -- one which is a concept, that is, a node somewhere else in the network (e.g., COLOR), and another which is a non-introspectable, primitive part of the notation itself.

Second, while a link like INSTANCE or MEMBER indicates something about the concept (or class) to which it is attached, <TELEPHONE COLOR BLACK> never means that the concept of a telephone is black. Property-asserting links at concept nodes indicate something about each of the members of the class, rather than the class itself.

Further, two deceptively different uses of attribute links are made at concept nodes***. To describe potential class members, we may wish to indicate a particular value for a given attribute that holds for every

* This connection is often acknowledged by authors, but usually superficially. It is most often unspecified how to make a concept node actually act as a relation between two other concept nodes -- nets are generally not implemented to facilitate such use of concepts (see [Shapiro 1971a, 1971b] and [Schubert 1976] for attempts at using concepts as relations). As a result, the "definition" of the relation is usually mnemonic rather than descriptive of how to use the relation under various processing conditions.

** This "triple" notation is the standard way of linearly capturing a link between two nodes. When expressing a property (attribute plus value) of some concept, it is to be read this way:

< concept node attribute value of attribute >.

Thus, < TELEPHONE COLOR BLACK >

says that the "color" attribute of the thing stood for by the node TELEPHONE has the value "black".

*** Woods points out this dichotomy in his "What's in a Link" paper [1975a].

Section 4.2.4
The trouble with properties

member of the class (e.g., <TELEPHONE COLOR BLACK> would mean that every telephone has the particular color, black). This is the use illustrated above, in Fig. 4.13; I will refer to this intent of a link as "assertional import" (following [Woods 1975a]). On the other hand, an attribute link at a concept node may be intended to describe the value rather than uniquely specify or name it. That is, one might wish to circumscribe a set of legitimate values to be expected as the value for the particular attribute in an instance, rather than demand the same value for that attribute in all cases. A "generalized property" could thus outline the general class of things to expect as the value of an attribute without dogmatically insisting upon a single value.

The dichotomy of intent between value specification and what I will call value description gives rise to an ambiguity in the more or less "standard" semantic net notation: in the case where the single value is specified, the intent is for each instance to manifest the particular property verbatim. This use, as mentioned, is often referred to as the "inheritance of properties, and usually results in identical notations for the representation of an inheritable property at the concept node (e.g., <TELEPHONE COLOR BLACK>) and the representation of the inherited one at the instance (e.g., <T1 COLOR BLACK>). Yet notice the subtle difference -- the former is asserting something implicitly about many individuals, while the latter is saying something explicitly about a particular one. That is, <x COLOR BLACK> is ambiguous! The former interpretation we might express as

FOR EVERY y / (INSTANCE/OF x) ; BLACK(y)

while the latter simply expresses the predication BLACK(x)*.

* The "FOR" notation is adopted from Woods [1968], and the example above should be read "for every y in the class (INSTANCE/OF x), assert the predicate BLACK to be true of y." The slash ("/") indicates the range of the variable (x), and the semi-colon (";") precedes the predicate to be asserted.

4.2.5. Implicit import of the class membership link

Even if we believe that the above problems can be gotten around, we are forced to acknowledge a final critical assumption. Since we intuitively feel that a concept captures what it means to be a particular kind of thing, the link that connects a concept node with a node representing an instance of that concept must pass on that definition to the individuating node. That is, by virtue of being an instance of the concept COMMAND, the PRINT/CMD must inherit a set of properties commonly known to be attributable to commands. This inheritance of the general properties and restrictions of a concept is implicit in semantic networks -- it just happens, without the mechanism for its happening being accounted for. This transmission of constraints and values from concept node to individuating node has three distinct aspects: 1) properties that are asserted at concept nodes with particular values will have those values directly inherited by each individuator (e.g., the intent of <LIST/CMD DESTINATION LINEPRINTER35> is for every invocation of LIST/CMD to have the particular value, LINEPRINTER35, as its DESTINATION), 2) restrictions of potential attribute values must be satisfied by particular values for the instance (e.g., the intent of <PRINT/CMD DESTINATION PRINTING/DEVICE> is that each PRINT/CMD instance have some device as its destination), and 3) special links (e.g., INSTANCE itself) must not be passed on at all. This gives the INSTANCE link, the purveyor of the inheritance, a complex meaning -- one which exists only by virtue of the special characteristics of the routines written to process the structure, and which is certainly not apparent from the notation itself. We must be very careful about links like INSTANCE, whose import is really a combination of several operations applied selectively to other links emerging from the All of the complexity (and thus the problems) gets buried in the processing routines, and becomes apparent only under close scrutiny.

4.2.6. The need for an epistemological foundation

This brief investigation into what nodes are supposed to stand for has shown that the foundations of semantic nets are not particularly sturdy. Let us recap: where does the prevailing view of "concept" leave us? For one thing, it seems to advocate defining a concept as an unstructured set of properties. It also leaves us with a very confused idea of how the links at a concept node contribute to the meaning of the node: we have a special link like INSTANCE, which takes its meaning from the way network-processing routines allow it to pass restrictions and values (and as I pointed out, "passing" itself is a complex operation); we have links intended to describe the properties that potential instances of a concept might have; and we have links that specify particular properties of particular entities. Yet each of these relationships is expressed in a superficially identical manner.

The differences in import of these links indicate that the uniformity of semantic network notation can be misleading. While it initially appears that we can simply map different kinds of relations directly onto links in the network, it turns out that very special machinery is needed to allow a system to make the "correct" interpretations of the links (i.e., the ones the designer intends).

It is the goal here to create a notation that avoids the fundamental difficulties discussed in this section. I believe that to overcome the expressive inadequacies of semantic nets, we must reevaluate our approach to the formalism at the foundational level. It appears that it is necessary to separate relations that are really primitive (i.e., that cannot be expressed other than circularly) from those that can be built from others (genuinely primitive link types require their own special-purpose interpreting routines, while others should be able to be interpreted using general-purpose programs). If we implicitly insist on primitive operations like description of potential instances, predication, and inheritance, then we must acknowledge that different

kinds of links must be used to implement these functions. It must be clear how to compose new relations from existing ones, and how in turn to apply such compound relations in general, rather than have processing routines anticipate every possible relation in advance.

In the next and final section of this chapter, I will reintroduce SI-Net notation as an intermediate level of structuring -- an epistemological foundation for representing concepts in semantic nets -- that explicitly accounts for the various operations assumed about network notations but generally obscured by the uniformity of those representations. Rather than force knowledge of the world directly into a simplistic node-plus-link system, I will use a set of epistemological primitives as a language in which the parts and features of concepts can be specified in a consistent and extensible way. It is this level of representation which would allow us to make the kinds of discriminations outlined above, and which is missing from common notations; this constitutes a significant change of approach to semantic network representations of knowledge.

4.3. Describing potential (and actual) instances

This section develops the basic set of primitives for describing the epistemological "parts" of concepts.

4.3.1. Binary relations for property description?

One attempt we might make to differentiate between the links used at concept nodes to define (describe) properties and those used at nodes for individuals to specify properties might be to create a primitive kind of link called, say, "HAS-AS-PART" (or "HAS-AS-PROPERTY"). Such a link could point from a concept node to a node that specified a class of legal values for the property, such as in <ARCH HAS-AS-PART BRICK>.

Unfortunately, this leaves it unclear as to which part or property of the arch it is that we are talking about. If bricks were always lintels, then we would have no problem, but arches can have other associated parts which may be bricks (namely, their uprights).

Perhaps, then, the HAS-AS-PART link should instead point to a relation, like LINTEL. This would make the concept specification look like a traditional case frame definition for a verbal concept, e.g.,

```
<SELL HAS-AS-PART AGENT>  
<SELL HAS-AS-PART OBJECT>  
<SELL HAS-AS-PART RECIPIENT>
```

.
.
.

But in this case, we are forced to believe that the class of values that can fill each role is fixed by the role itself, e.g., that every AGENT is a PERSON. Unfortunately, this leaves us with an overly rigid notion of case -- as Schubert [1976, p. 169] points out, ". . . the domain restrictions associated with a particular case vary from one predicate to the next." The notion of a case with a fixed class of values across all concepts is not sufficient to distinguish between two distinct aspects of cases, namely, the value class and the functional role. We could not realistically expect to limit the possible fillers of general cases like OBJECT or INSTRUMENT across all concepts without reducing them to trivial placeholders.

4.3.2. Describing attribute complexes -- the basic notation

It is clear that the definition of a part or attribute of a complex concept requires more than a simple binary relation. First, we need some indication of the class of entities from which a legal value may be drawn. This part of the attribute definition describes, in a structural way, the potential case fillers. Second, we would like a way to indicate the relationship that the value will have to the structured

object as a whole -- this is the relationship that is usually indicated by naming the link. This second aspect of the attribute (or part) definition tries to capture the functional role to be played by the particular piece of the entity. Note that the standard <concept attribute class of values> triple attempts to include both role (attribute) and value class information in a single binary link. However, as we have seen, this is not a satisfactory notation, in that the net-processing routines must do something special to differentiate between the meaning of "attribute" in the above and in <individual attribute particular value> (see [Woods 1975a, p. 70]).

I would like to take an approach to networks that separates out explicitly all fundamental operations, so that each of the critical underlying mechanisms of the network will be available as a primitive. The links that carry out each of these foundational representational duties will be the epistemological primitives out of which more complex conceptual structures can be constructed. Here, then, are isolated three operations that are confounded by typical semantic net notations -- the description of a generalized attribute (or part) to be found with each instance, the value class of legal fillers for that attribute, and the functional role that the defined part fulfills. Let us reflect each of these in primitive link types in the foundational Structured Inheritance Network notation*:

-- DATTRS

points from a concept node
to a role description node

interpreted as pointing from a node which implicitly defines a class of objects of the world being represented (by virtue of the node's being interpreted as a predicate true of each of those objects individually), to a node which describes a "dattr" (attribute or part) of each of those objects.

* It is not necessary that these be the only (or correct) ones -- the idea of a DATTRS-like link to a descriptive node is what is important here.

-- VALUE/RESTRICTION (V/R)

points from a role description node
to a concept node

the role description node from which this link emerges is the description of one of the functional parts of an instance of the concept being defined (the node connected by a DATTRS link to that role description node). The VALUE/RESTRICTION link points to a concept that circumscribes the class of entities that can be considered to play the part in the instance.

-- ROLE and ROLENAME

points from a role description node
to a role description node (ROLE)
or a string (ROLENAME)

specifies the functional role to be played by the part being described at the source role description node. If ROLE, defines the current role to be inherited from the role description node pointed to. If ROLENAME, the string is considered to be the name of the role, and no further roles are accessed.

Notice that we need an intermediate node to hold the VALUE/RESTRICTION and ROLE pointers. Such a node is called a "role description node", and it is to be a place which embodies the definition of an important functional part of the kind of entity being defined by the concept of which it is a part. The abstract entity represented by such a node I will call a "dattr", for "description of an attributive part". A role description node may be labelled (i.e., the ROLENAME link indicates a string) -- in which case the role specified by the label is thought of as being defined at that particular node -- or the ROLE pointer may indicate a more general role description that includes the current one (I shall return to this below).

Figure 4.14 illustrates the basics of this role-oriented attribute-description mechanism. In this figure, the role description node S will

* While we can easily talk about the potential role fillers or the role itself, there is currently no existing terminology that captures the combination of a filler playing a role in a complex. It is to fill this gap that the term "dattr" has been invented.

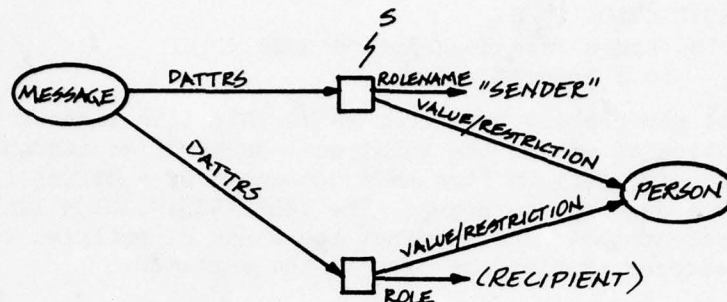


Figure 4.14. Basic SI-Net notation.

constitute the definition of what it means to be a sender of a message (in Chapter 5 I will illustrate how this definition is carried out). The node MESSAGE will then be the implicit definition of a set of entities which satisfy the criteria defined by the dattr descriptions. Note that this node stands for "the generic concept of a message", that is, a singular entity of some sort, rather than an entire set. In that case, each role description should be mapped directly onto a corresponding role filler for each node representing an individual. The concept node defines in this way a predicate which is true of all messages, and only implicitly defines a class of objects*.

Given that we now have a place to access the definition of a kind of a part of an entity, we find that this is a good place to associate other useful information. In particular, from a role description node we would see emanating the following links:

-- NUMBER

points from a role description node
to a number predicate

this link indicates the number of fillers of the source role to be found in an instance. For example, in the figure above, it should not be necessary to restrict a message to having a single recipient.

* It has, in fact, been suggested by David C. Brown [Brown & Kwasny 1977] that the set and its membership relations be explicitly represented, separate from the concept node.

Section 4.3.2
Describing attribute complexes

-- MODALITY

points from a role description node
to a modality

some attributes of a particular type of thing may be important, but not necessary. Others, like the vertical clearance of an ARCH, are derived from the structure of other dattr's. The MODALITY link allows the role to be interpreted as an optional or derived one. Valid modalities are NECESSARY, OPTIONAL, and DERIVED*.

Figure 4.15 presents a more complete use of this notation. In this

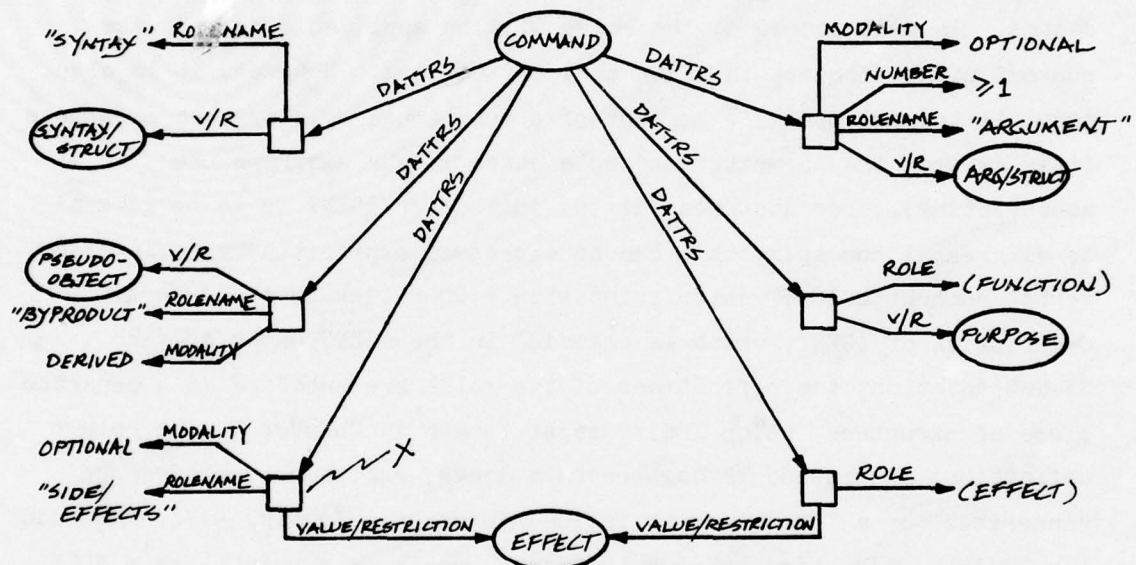


Figure 4.15. A node for COMMAND.

* After considerable contemplation, I have found this distinction not subtle enough. First, "NECESSARY" ignores the difference between "obligatory" roles, whose fillers are critical to the identity of an individual, and "inherent" roles, whose fillers are guaranteed to exist but whose values are not critical. Second, "DERIVED" should perhaps be "DERIVABLE", since fillers are not always derived. In addition, DERIVED is not mutually exclusive with the other modalities, but can be specified as well as, say, OPTIONAL.

figure we see a node (COMMAND) which defines six criterial attributes or parts for commands. The VALUE/RESTRICTION link for the SIDE/EFFECTS dattr (represented by node X) points to EFFECT, which is itself a concept. Any particular individual EFFECT is therefore a legal filler for the SIDE/EFFECTS slot of a particular command. Further, SIDE/EFFECTS are optional (a thing can still be a command if it has no side effects).

Note that dattrs provide a generalized case definition facility -- cases are defined relative to the particular concepts of which they are dattrs. In most concepts, the roles will be applicable only in the context of the concept in which they are defined. However, it is also possible to make use in a concept of a functional role defined elsewhere (this is what the parenthesized role names in our diagrams are abbreviating). For instance, if the idea of an AGENT is to be general to all verbal concepts, this can be expressed explicitly by having each verbal concept's AGENT dattr point with a ROLE link to the general description of AGENT, which is embodied in the AGENT dattr of VERB. In SI-Net notation, the definitions of the roles are embodied in a separate piece of structure, which I discuss at length in Chapter 5. The role definitions are passed through certain links, and thus a role can be "inherited" by a lower concept in the hierarchy. In Fig. 4.16, the node for TO/SELL, a particular verbal concept, needs only point with a ROLE link to the parent concept's role description node to inherit the definition of an AGENT, which is part of the definition of TO/SELL. This is because of the special properties of a link called "DSUPERC", which I discuss later. If, on the other hand, a concept such as LINTEL is only meaningful in the context of ARCH, the dattr node for LINTEL in the ARCH concept will itself constitute the definition of that role, and will not point to any more general definition. Instead, a string will indicate the name of the role represented by the node, and the definition of a LINTEL will be specified completely by the role defining structure of ARCH (and not inherited from some more general concept). I

Section 4.3.2
Describing attribute complexes

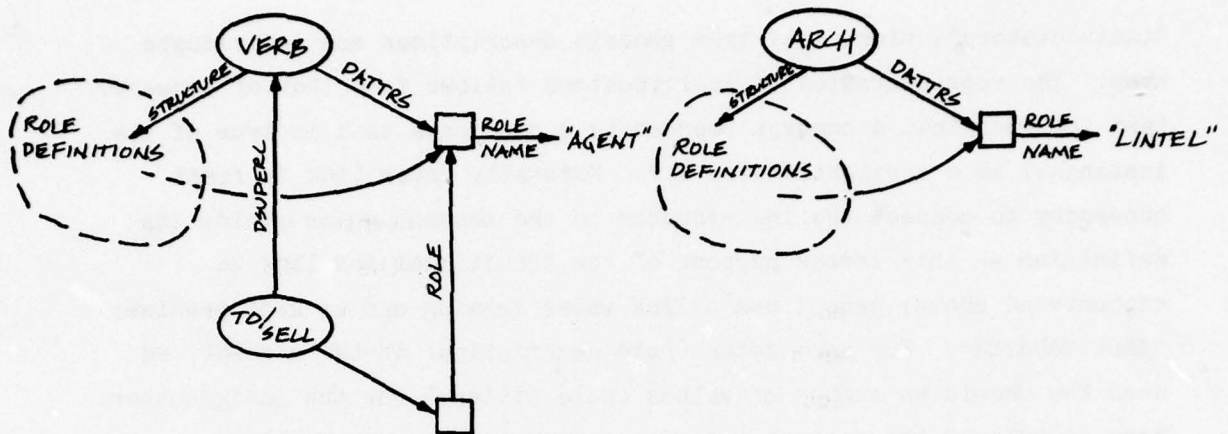


Figure 4.16. Roles and role chains.

will say more about roles as places in structural complexes in Chapter 5.

4.3.3. Individuators

Now that we have explicitly broken out the definitions of parts at concept nodes, we need to reevaluate the way that instances can be described. As mentioned earlier, in standard nets we usually find assertions about the properties of particular objects represented by attribute/value pairs attached to nodes for those objects. Woods [1975a, p. 53] points out how attribute links at an "instance node" in many notations are used mistakenly to indicate "arbitrary relations" that exist between the instance and certain values. Given the way I have broken apart the parts of a concept, I would have to agree with Woods that this should not be the case: the concept to which the instance node is attached defines a set of criterial roles that must be filled to make a well-defined instance. The attribute links that can appear at descriptions of instances are contextually determined, and are not so arbitrary.

The nodes that are used to represent instances are called

"individuators", since they take generic descriptions and individuate them. The representation of individuators follows from that of concepts (and the fact that a concept represents a predicate that is true of its instances) in a straightforward way. Naturally, some link is first necessary to connect the individuator to the concept which yields its definition -- this is the purpose of the traditional ISA link we encountered above; here I use a link whose meaning can be kept precise: "INDIVIDUATES". For each dattr (role description) of the concept, we need the requisite number of values (role fillers) for the individuator. This is because the concept node is constructed as a generalized singular description, with a DATTRS link for each role to be filled in every instance. If the concept is considered to be a predicate or function, then each of the non-DERIVED role fillers is an argument.

An "attribute/value" pair is then represented as in Fig. 4.17. The

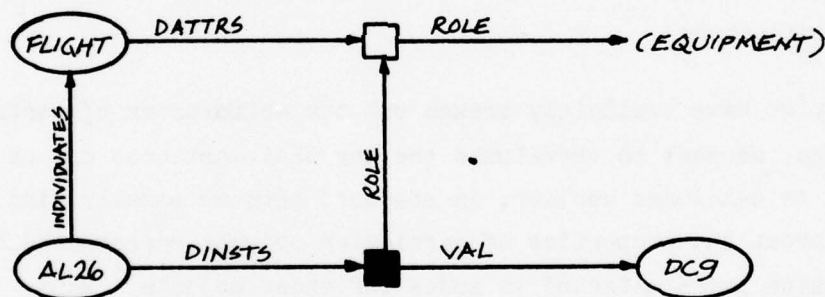


Figure 4.17. An "attribute/value" pair.

vertical ROLE link indicates the appropriate "attribute" (EQUIPMENT) to which to bind the value (DC9), which is in turn indicated by the VAL link. The DINSTS link constitutes the explicit statement of the assertion, "the EQUIPMENT of FLIGHT AL26 is a DC9." I will henceforth refer to nodes pointed to by DINSTS links as "role instance" nodes; these nodes will be shaded in diagrams. Note that, in the spirit of explicit description, I have made clear the meaning of the "attribute/value" pair in terms of its underlying foundational operations, definition and binding.

4.3.4. A note on the standard abbreviated notation

Before moving on to the exegesis of the rest of the formalism, let us pause to make an observation about the more common network notations. Bearing in mind our explicit representation in terms of epistemologically primitive relations, we can see that the most commonly used representation of an attribute of an object is really an abbreviation (see Fig. 4.18).

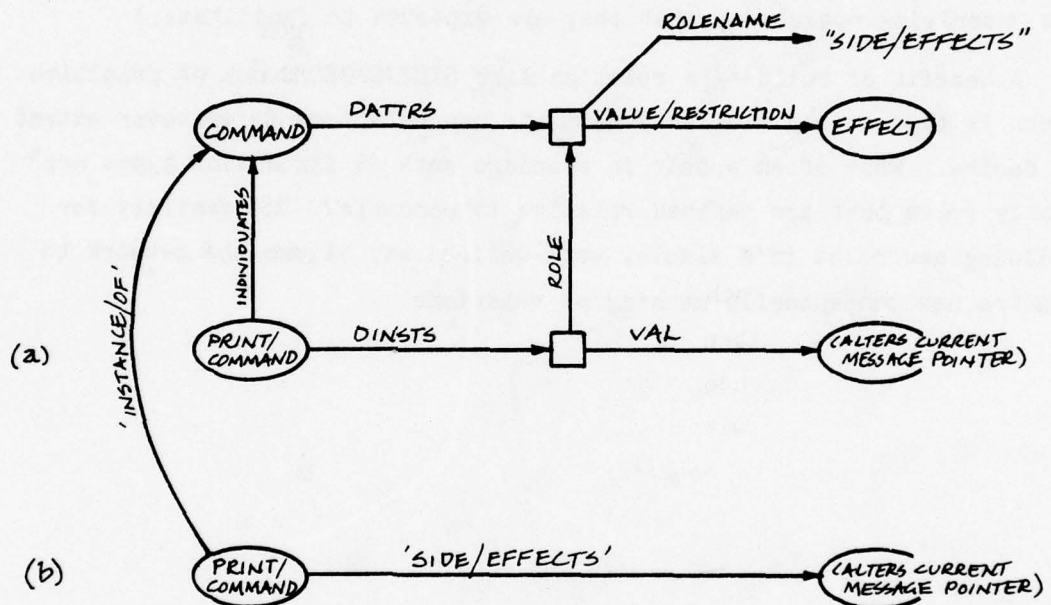


Figure 4.18. Conceptual relations.
(a) unabbreviated
(b) abbreviated

In Fig. 4.18, (b) appears to be an abbreviation for the more complex structure exhibited by (a). However, Woods [1975a, pp. 69-70] has pointed out that this network notation is also used to express the "constraints on the possible fillers for the arguments of the [concept] predicate", thereby having "the same link names meaning different things depending on the nodes which they are connected to". This common difficulty is resolved when we explicitly distinguish between the two

underlying epistemological operations. In other words, we use the DATTRS and VALUE/RESTRICTION links to indicate constraints on possible fillers (what might normally appear as <COMMAND SIDE/EFFECTS EFFECT>), and the DINSTS and VAL links to indicate particular fillers (what might normally appear as Fig. 4.18(b)). Thus, a mechanical procedure that operates on network structure will not be confused by an ambiguous link name like "SIDE/EFFECTS". (This is a good example of how we can better understand problems with network notations by trying to make explicit the underlying operations that they are expected to facilitate.)

A benefit of building a relation like SIDE/EFFECTS out of primitive links is that we can similarly generate new relations to whatever extent we desire. What often appear in standard nets as fixed link types are really roles that are defined relative to concepts. The facility for building new roles in a simple, well-defined way allows the network to acquire new conceptually meaningful relations.

Chapter 5. What Holds Things Together? -- Intensions and Structure

In the last chapter I began the presentation of a set of epistemological primitives for building descriptions of classes of instances. Here I take a closer look at what it is that we want to build from systems of those primitives. I have so far operationally defined a "concept" as an SI-Net entity which describes all of the potential members of the class, and a "concept node" as a notation for a concept; I proposed thinking of these concepts in two ways to accomplish this: 1) as predicates (or functions) which apply to all instances, and 2) as sets of generalized attribute descriptions (dattr) which must be instantiated to describe an instance. In this chapter I shall attempt to integrate these two views by using the classical notion of intension.

Section 5.1 will introduce the idea of an intension, and will show how the concepts developed in Chapter 4, if viewed as representing the intensions of predicators, lack an essential ingredient -- the body of the predicate. I will show how the SI-Net formalism allows a structured representation of the body and how it attempts to capture how this crucial element relates to the definitions of the roles associated with a concept.

In Section 5.2, we move on to the derivation of new concept-level units from existing ones. Here a well-defined foundational notation proves invaluable in making concept-derivation facilities general. Finally, we will look at some of the more general implications that intensions have for a knowledge representation scheme. I will briefly illustrate how this notion allows us to achieve some precision in the definitions of semantic network nodes.

5.1. Intensional structure

If our net structures were intended to capture only the class membership and subset relations, we might say that the representation was purely extensional -- that two nodes represented identical concepts if they both had the same members. But, as we have seen, we invariably want these class nodes to represent "concepts", which are to capture more than just class membership. Concepts require a mechanism for describing their extensions (i.e., both existing and potential members of the set), and thus are not identical unless the descriptions that those concepts embody are the same. (That is, if the properties of the potential instances of two concepts could be shown to be the same through the logical structure of the network, we could predict that the extensions would be equivalent in any possible world to which the descriptions were applied.) Thus, by interpreting concepts as descriptions of potential instances, we can determine and make use of the relationships between them independent of the particular objects to which they apply. It is this non-extensional ("intensional") use of the semantic network which should afford it so much expressive power. But how is the notion of an intension to be captured in a network notation?

5.1.1. Intensions and concepts

First, what is an intension, and why might that notion be useful here? Much of the philosophy of language has been dedicated to the attempt to formalize the intuitive idea expressed above about "what it means" to be something. Philosophers associate with language expressions ('designators'*) two types of abstract entities -- an

* I will use the following notions, from Carnap [1947, pp. 6-7]: "I propose to use the term 'designator' for all those expressions to which a semantical analysis of meaning is applied . . . Our method takes as designators at least sentences, predicators (i.e., predicate

extension, or the entities of a particular world designated by the expression, and an intension, an abstraction of the properties of those individuals which acts in such a way as to select from any possible world the set of individuals that are described by the language expression. For example, take the case of a predicate expression -- the kind of natural language designator that corresponds to the interpretation of concepts as predicates that I have been advocating. Carnap states that "the intension of a predicator (of degree one) is the corresponding property" [1947, p. 19] (as opposed to the class of things that possess that property). Woods characterizes the intension of RED this way [1975a, p. 49]: "... an abstract entity which in some sense characterizes what it means to be red, it is the notion of redness which may or may not be true of a given object . . ."; and more generally: "In many philosophical theories the intension of a predicate is identified with an abstract function which applies to possible worlds and assigns to any such world a set of extensional objects (e.g., the intension of 'red' would assign to each possible world a set of red things)."

Thus, the intension of a predicator can be thought of as an abstract entity that somehow expresses what it means to satisfy the predicate, i.e., the abstract "property" itself. Carnap extends this notion to define relations, functions, and individual concepts, which he eventually classes (along with properties) under the term concept. Relations and functions are the intensions of predicators of degree greater than one, and functors, respectively, and reflect the general use of those terms. Individual concepts are the intensions of expressions that attempt to designate particular individuals in the world of discourse (functions of degree 0). This latter category includes names as well as expressions built from more general terms,

expressions, in a wide sense, including class expressions), functors (i.e., expressions for functions in the narrower sense, excluding propositional functions), and individual expressions . . ."

prefaced by "the x such that . . .".

Thinking back on the class/concept difference remarked upon in Section 4.2.2, we find that our initial confusion stemmed from a lack of appreciation of the intensional nature of semantic net representations. There is a certain class of operations that we expect to do on representations of knowledge that involve definitional connections between concepts. It is this intensional side of concept nodes that would allow a program to determine what relationships were entailed by the assertion of a particular relation, or when to assert a relation given others, or how to assimilate (build in terms of the others) new conceptual units. In short, how the definition of a conceptual entity is derived from and related to all other conceptual entities is the key to making intelligent use of the knowledge embodied in the net. This is the kind of use Quillian originally envisioned for his nets (in TLC, in particular), but was never able to realize. The formal notion of intension is precisely what is needed to firm up our representations enough to perform that kind of task.

While philosophy texts never really tell us what a primitive intension looks like, semantic nets provide us an opportunity to examine a potential representation scheme for all kinds of intensional entities. Not only can we experiment with inter-intensional connections, we can deal with the internal intensional structure of a single concept as a manipulable entity. Semantic network nodes can be considered representations of the intensions of natural language designators (namely, predicates, functors, and individual expressions)*.

* Woods [1975a, p. 53] uses the EGO link to represent the intension of a node. What I will try to show here is how, in a well-defined net, the intension can be derived directly from the constellation of links impinging on the node.

5.1.2. The body of the predicate

If we look at the constellation of links around a typical concept node as representing the intension of a predicator (or functor), we can immediately apprehend the lack of a critical piece of information. How do the properties and parts that describe a potential instance go together? Most often we see concept nodes with links that describe "arguments" to the predicate (or function) which the concept expresses -- for instance, a node for the concept of an ARCH will have what amounts to an argument for its lintel and one (or two) for its uprights. But there is no procedure or predicate "body" to express what makes these three things into an arch, rather than a set of three bricks*. (We note, for example, that a stack of some objects is not the same as a pile of the very same objects.) It is how the parts and properties go together -- the "gestalt" -- that is required to make the concept description complete.

What I am saying here is that a concept node cannot express "what it means" to be an ARCH by describing the parts and properties of potential instances of ARCH independent of one another. The basic SI-Net notation described in Section 4.3 gives us only the generalized definitions of the arguments to the concept predicate. Thus a concept node, as I have presented it so far, resembles a programming language function, with a header that shows the formal parameters -- but one with no body. In the terms of this report, more than just dattr's are necessary to represent concepts.

In SI-Net notation, we can provide a way to include this structural pattern in the concept definition. A primitive link type called "STRUCTURE" ties a concept node to a structure representing the

* This problem is clearly illustrated by "case frame" representations of concepts. Knowing that we can have an animate AGENT, an inanimate OBJECT, etc. juxtaposed gives us no clue to the actual process which takes place between them.

interrelations among its roles. For each concept, one STRUCTURE link emerges from the concept node; this link points to a group of tokens of other concepts in the network, which refer appropriately to the dattr of the concept being defined. This group of tokens is called the "structural condition" (although, as mentioned in the footnote in Section 4.1, we might consider multiple structural conditions).

For example, the structural condition of a node for the noun-noun compound hydrogen bomb might express the reaction that occurs between the elements of the bomb in terms of high-level concepts like EXPLODE, NUCLEAR/CHAIN/REACTION, and FUSION, as in Fig. 5.1. In this figure, there is a STRUCTURE link from the node which represents the bomb to node E, which specifies that the bomb EXPLODES by virtue of the reaction specified by node F. (Notice that in order to indicate how the entire bomb, as an entity, explodes, we make use of the dattr whose role is "WHOLE".) Node F in turn expresses the fusion reaction of the HYDROGEN part of the bomb as caused by the chain reaction represented by node N.

Note that these nodes constituting the structural condition (i.e., E, F, and N) look very much like the individuating nodes that I illustrated in Section 4.3.3, except that COREFVAL links from their role instance nodes point to role descriptions of the enclosing concept, and not to other concepts. These links are very special -- they point to the "insides" of concepts rather than to concepts themselves, and represent intensional ties between potential fillers of roles. Remember that concept nodes like the HYDROGEN/BOMB in Fig. 5.1 represent patterns to be fit by all particular instances of the concepts. In the SI-Net version of a pattern, a role description node is the description of all of the potential fillers of a particular role, and a "paraindividual" node in the structural condition schematically represents a relationship which will apply between the particular fillers in any instance of the concept. This means that a COREFVAL pointer from a structural condition node (e.g., node F1 in the figure) to a role description node (e.g., node H) is to be interpreted as accessing not the role description

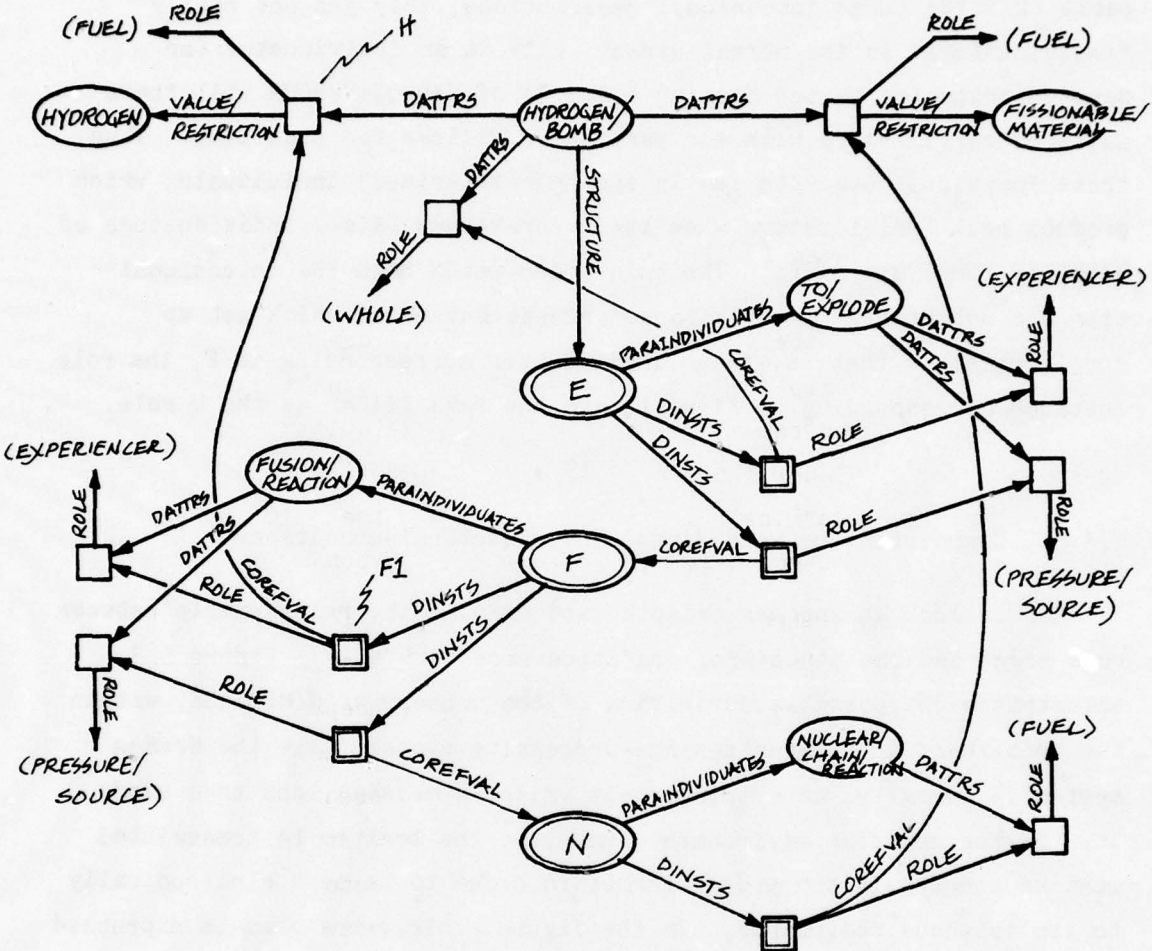


Figure 5.1. HYDROGEN/BOMB in terms of other concepts.

itself, but its ultimate filler when the concept is applied (just like the expression "for all x , $f(x)$ " does not mean to apply the function f to " x ", but to each value of the variable selected by the quantifier). As I mentioned earlier, intensional connections are those that appear regardless of the particular extensional entities to which they apply. Thus, since ties between nodes like F1 and H indicate relations true of all entities to which the concept applies, they are intensional, in the sense that they define the structure of the extensions to which they apply. And since token nodes like E, F, and N in Fig. 5.1 have as their

parts (DINSTS) these intensional descriptions, they are not really "individuators" in the normal sense. Only in an inductor (or particularization -- see Section 5.2.1.3) of HYDROGEN/BOMB will these patterns be filled in with the particular fillers for that bomb. Thus, these individual concepts really are "parameterized" individuals, which produce real individuators when their parameters (i.e., individuators of HYDROGEN/BOMB) are given. The role nodes which hold the intensional ties are not really role "instance" nodes, but nodes which set up coreferences -- that is, in an inductor corresponding to F, the role instance corresponding to F1 will have the same filler as the H role.

5.1.3. Connections between dattr and structural condition

Let us look at another example, and examine the relationship between role nodes and the structural condition more carefully. Figure 5.2 illustrates one possible definition of the concept of a MESSAGE, within the context of a computer message-processing system (like the Hermes system). Normally, an author merely writes a message, and then sends it. In the computer environment, however, the text to be transmitted must be encoded in the proper format in order to be sent electronically to its intended recipients. In the figure, this extra step is expressed as a function, the RESULT of which is operated on by a "SEND" action (in the simpler case, the text itself would be directly sent). Thus, the structural condition of MESSAGE has two paraindividuals, with the MESSAGE role of (SEND) being filled by the RESULT of (ENCODE)*. The RECIPIENTS of a MESSAGE are the same entities as those considered the

* Notice that in this figure there is a role with the same name as a concept (MESSAGE). This kind of duality is common in natural language, and while it often causes confusion in the meanings of sentences, it lets us be appropriately ambiguous when necessary. SI-Net representation gives us at least the facility to represent distinctly the different interpretations.

Section 5.1.3
Dattrs and structural condition

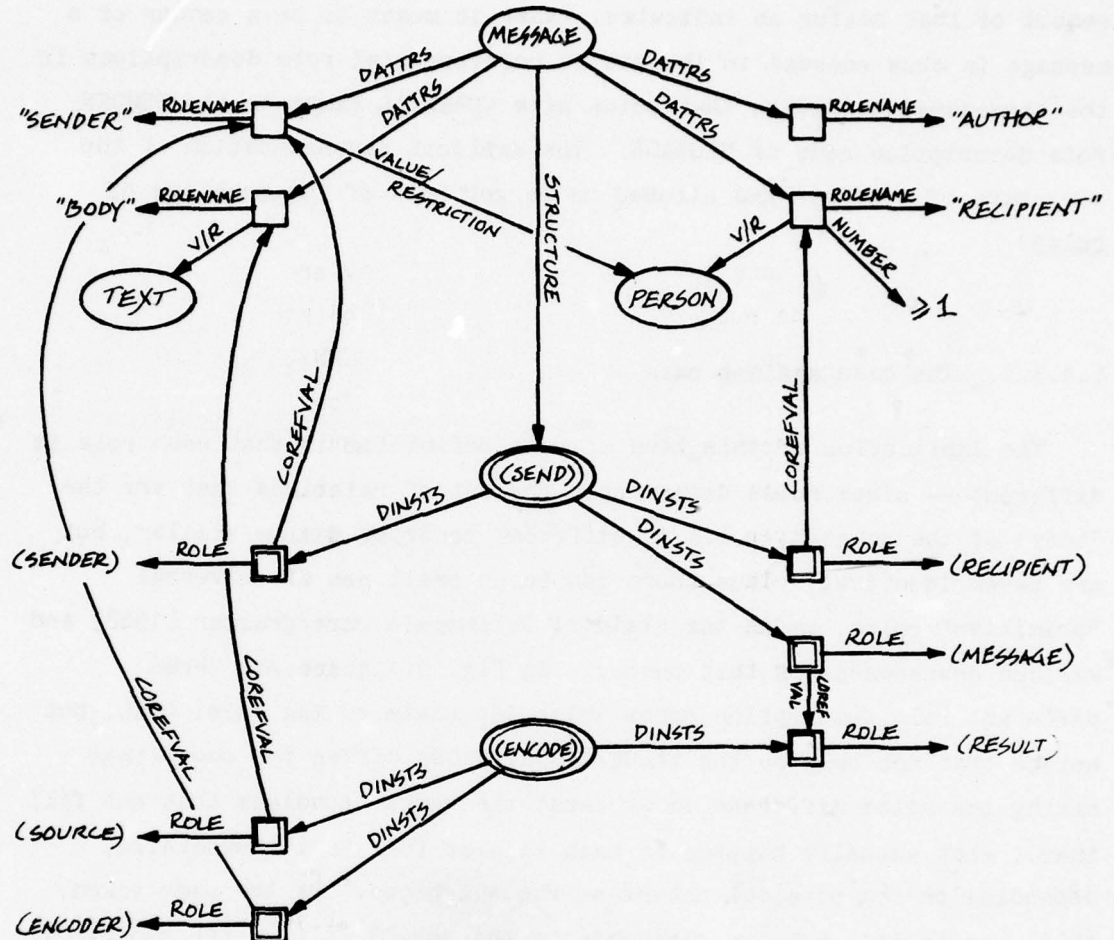


Figure 5.2. A MESSAGE in terms of operations on it.

RECIPIENTS of the SEND operation*.

In the case of the SENDER in this figure, we can see how the structural condition encodes the definition of the role. A role is defined by how its potential filler participates in the relations specified in the structural condition. To be a SENDER of a MESSAGE is to initiate the kind of ENCODE action specified, and then to SEND the

* The AUTHOR's role in the message is not yet accounted for -- I return to this in Section 5.1.4.

result of that action as indicated. What it means to be a sender of a message is thus encoded in the set of coreferential role descriptions in the structural condition that point with COREFVAL links to the SENDER role description node of MESSAGE. The explicit representation of the structure of concepts has allowed us to get hold of the meanings of roles.

5.1.3.1. The case against case

The implication of this kind of role definition is that each role is different -- since roles depend upon the set of relations that are the "body" of the concept, roles in different concepts may be similar, but are never identical. Thus there can be no small set of universal "primitive" roles, as is the claim of Fillmore's case grammar [1968] and various descendants of that theory. In Fig. 5.1 there are three different role description nodes which lay claim to the role, FUEL, but notice that not only do the VALUE/RESTRICTIONS differ for each (thus making the roles different in at least the range of values that can fill them), what actually happens to each kind of fuel is idiosyncratic, depending on the physical nature of the substance. By the same token, there are in Fig. 5.2 two claimants to the SENDER role -- the aspect of the message and the agent of the SEND operation. While the definition of the former includes that of the latter (in the sense that the SENDER of a MESSAGE, among other things, is the AGENT of SEND), the SENDER of a MESSAGE by our definition is more than just the initiator of the sending.

This is not really a new observation -- at least three criticisms of the notion of a small number of universal "cases" in semantic representations have appeared recently. Schubert [1976, p. 168] explains how "the notion of case derives from the systematic similarities between the roles played by the arguments of many predicates in relation to those predicates," and goes on to expose how

Section 5.1.3.1
The case against case

such perceived similarities are not representable identities. He examines several representations and shows how at least the domain restrictions associated with particular cases vary from predicate to predicate; he concludes with, "To my mind, a genuine understanding of the structured analogies between different sorts of actions requires analysis of such actions in terms of more elementary events" [1976, p. 169]. What has been provided here, beyond essentially the same observation as Schubert's, is a clear picture of the source of the problem (where he says "systematic similarities", SI-Nets have a way of representing them), and SI-Nets provide a particular kind of analysis of actions (and objects, etc.) in terms of more elementary pieces.

Charniak shows how current AI systems (notably those of Schank, Norman and Rumelhart, and Wilks) fail to describe the meanings of cases beyond "intuitive" ones, and how ". . . there is never any indication of how such intuitive meanings are to be coded into the system, or what the precise definitions are" [1975, p. 12]. I agree with his critique, and offer structural conditions and their relations to dattrs as a method for capturing precisely the definitions of cases. However, in accordance with Schubert, one should not expect the number of such case definitions to be necessarily small. Finally, Cercone [1975, p. 80] refers to an earlier discussion by Bartsch and Vennemann [1972]: "'The meaning' of an argument as argument is entirely determined by the relation. Therefore, no two arguments have precisely the same meaning, as arguments.'" SI-Nets allow us at least to capture the "meanings of arguments" as they are determined by relations (and functions, and objects).

5.1.3.2. Full roles

Despite the differences between them, it does feel natural to refer to the different types of fuels in Fig. 5.1 as "FUELS". In SI-Nets, the attempt is to capture the similarities between these non-identical roles by pointing from their role nodes to the general "FUEL" dattr. However, the real functional role of each of these DATTRS is a composite of the general idea of a fuel and the particular variation on that theme indicated by the local structural condition. Thus, node H's full role is something like "FUSIONable FUEL", while the other DATTR of HYDROGEN/BOMB has as its full role "NUCLEAR-REACTABLE FUEL". The full role is the combination of the destination of the ROLE link and the particular structural condition's use of the role description node.

The two idiosyncratic HYDROGEN/BOMB notions of FUEL are locally defined, and are meaningful only in the context of that concept (as is the notion of FUEL for NUCLEAR/CHAIN/REACTION). ROLE links to FUEL tie these locally defined notions to a more general notion. However, looking at Fig. 5.3, we see that none of these satisfies literally the more standard definition of FUEL, "a thing which is burned to produce energy". Yet while these roles on the surface appear disparate, there is a strong common sense between them. I believe that this may be somehow abstractable from the structural conditions, to create a more general definition for the role. For example, BURN may be generalized to some energy-liberating process that would allow inclusion of fissionable and fusionable substances as its FUEL. While I do not as yet have a proposal on how this might be done, it does appear to be a fruitful research area. (For now, as I have said, I will rely on the ROLE links to suggest the conjunction of the general role description and the particular local variant.) In any case, SI-Net notation differs significantly from ones that use role names merely as convenient names for slots, in that it provides substance (the structural condition) to role definitions.

Section 5.1.4
More complex S/C's

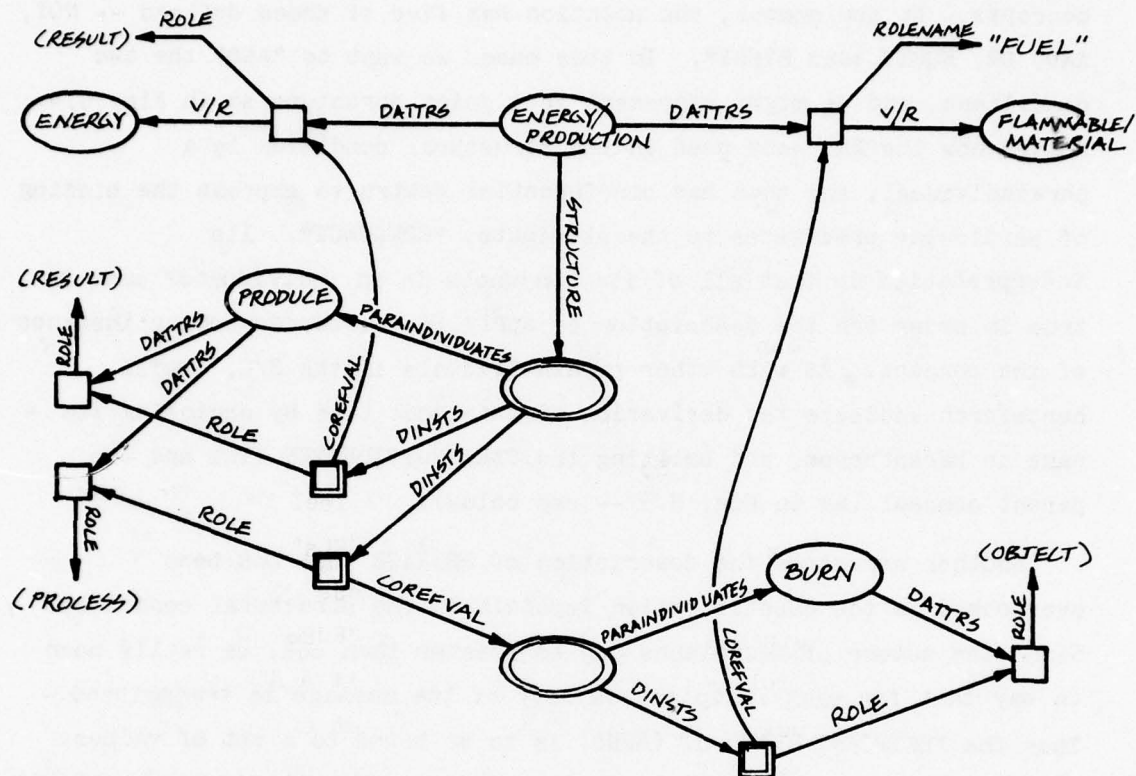


Figure 5.3. Basic definition of FUEL.

5.1.4. More complex structural conditions

In my description of a message (Fig. 5.2), I neglected to define what place the AUTHOR had in the structural description. Let us say that it is the author's role to compose the text that is to be encoded and sent. Such a description would be relatively independent of the relationships expressed in the structural condition in Fig. 5.2 -- while the SEND operation is dependent upon the result of ENCODE, neither is really dependent on the author's relation to the BODY (the BODY is an independent entity in this characterization, while the datatr for the RESULT of ENCODE is DERIVED). Thus we need a way to conjoin the two descriptions.

Conjunction of this type is accomplished by using primitive

concepts. At the moment, the notation has five of these defined -- NOT, AND, OR, EQUIV, and EVERY*. In this case, we want to "AND" the two conditions, and we might represent this joint structure as in Fig. 5.4. Notice how the AND node used in the structural condition is a paraindividual, and thus has coreferential dattr to express the binding of particular predicates to the attribute, "CONJUNCT". Its interpretation is that all of its conjuncts in an individuator must be true in order for the description to apply to the corresponding instance of the concept. As with other paraindividuals in the S/C, I will henceforth indicate the derivation of this node type by enclosing its name in parentheses, and omitting the PARAINDIVIDUATES link and the parent concept (as in Fig. 5.5 -- see below).

Another aspect of the description of MESSAGE that has been overlooked is the quantification implicit in the structural condition. Since the number of recipients can be greater than one, we really mean to say that for each recipient, a copy of the message is transmitted. Thus the RECIPIENT dattr of (SEND) is to be bound to a set of values. In many cases, it will be necessary to make such quantification explicit (e.g., to indicate precisely the scope of the quantification), and this is the intent of the EVERY node. EVERY has three roles: x, which specifies a class over which the quantification is to range; R, an optional predicate which further restricts the range of the quantification; and P, the proposition being quantified. This type of node reflects directly Woods' query language [1968], and its intended effect is precisely the interpretation to be given to the expression

FOR EVERY x / CLASS : R(x) ; P(x) .

In the example, the quantification of SEND is to range over all values of the RECIPIENT role; there is no further restriction on that class of

* Diamonds are used to indicate these concepts which do not have S/C's of their own. This is merely a notational convenience, since for all intents and purposes, primitive concepts act like normal ones.

Section 5.1.4
More complex S/C's

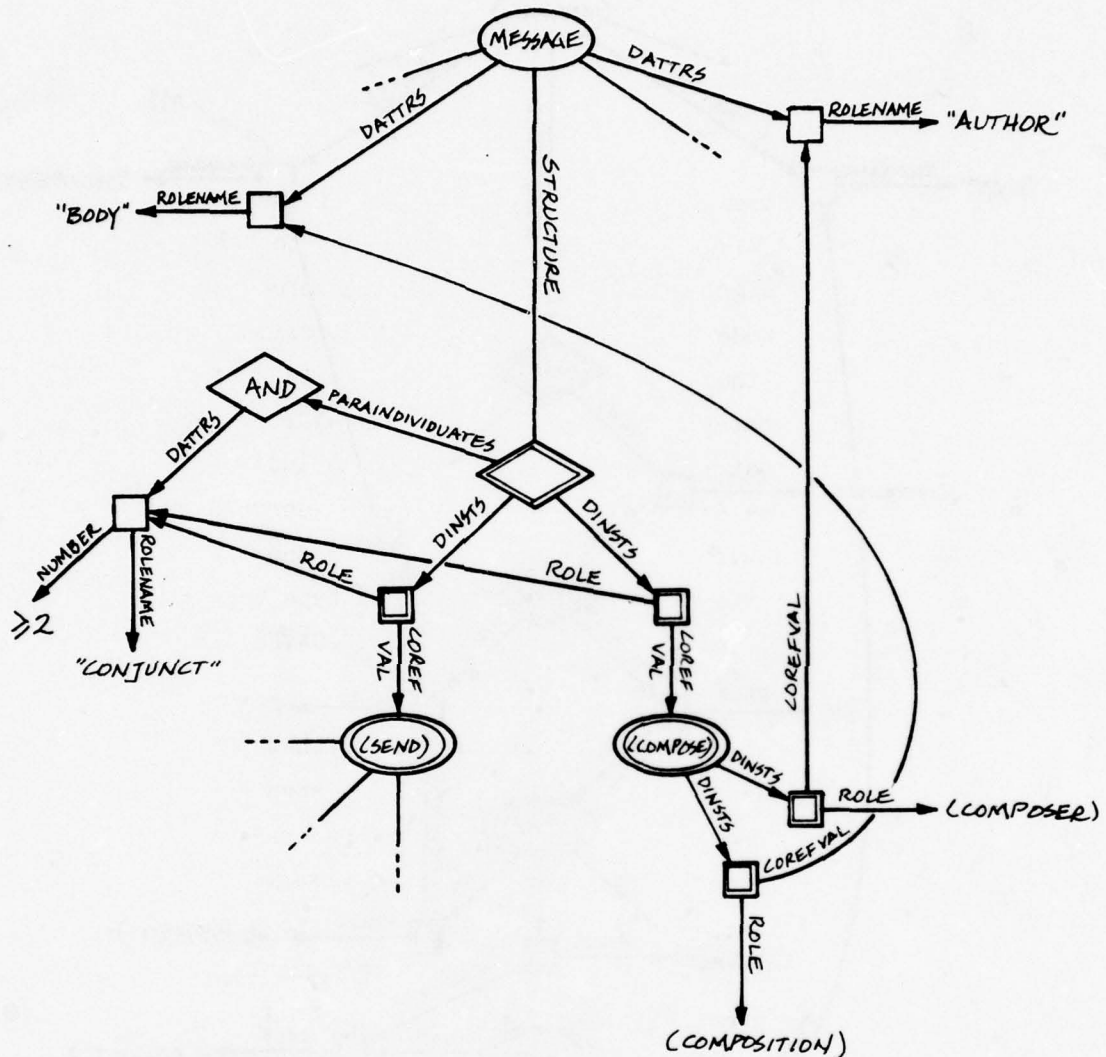


Figure 5.4. Conjunction in the S/C.

values. Fig. 5.5 details the connections between the RECIPIENT node, the SEND paraindividual, and the paraindividuator of EVERY. The variable, x , is to range over the set of RECIPIENTS of the MESSAGE, and this is indicated by a COREFVAL link from node E1 to node M1. The quantified predicate is (SEND); this is indicated by a COREFVAL link from the "P" role instance node of (EVERY) and the connection of the RECIPIENT of (SEND) to the "x" dattr of (EVERY).

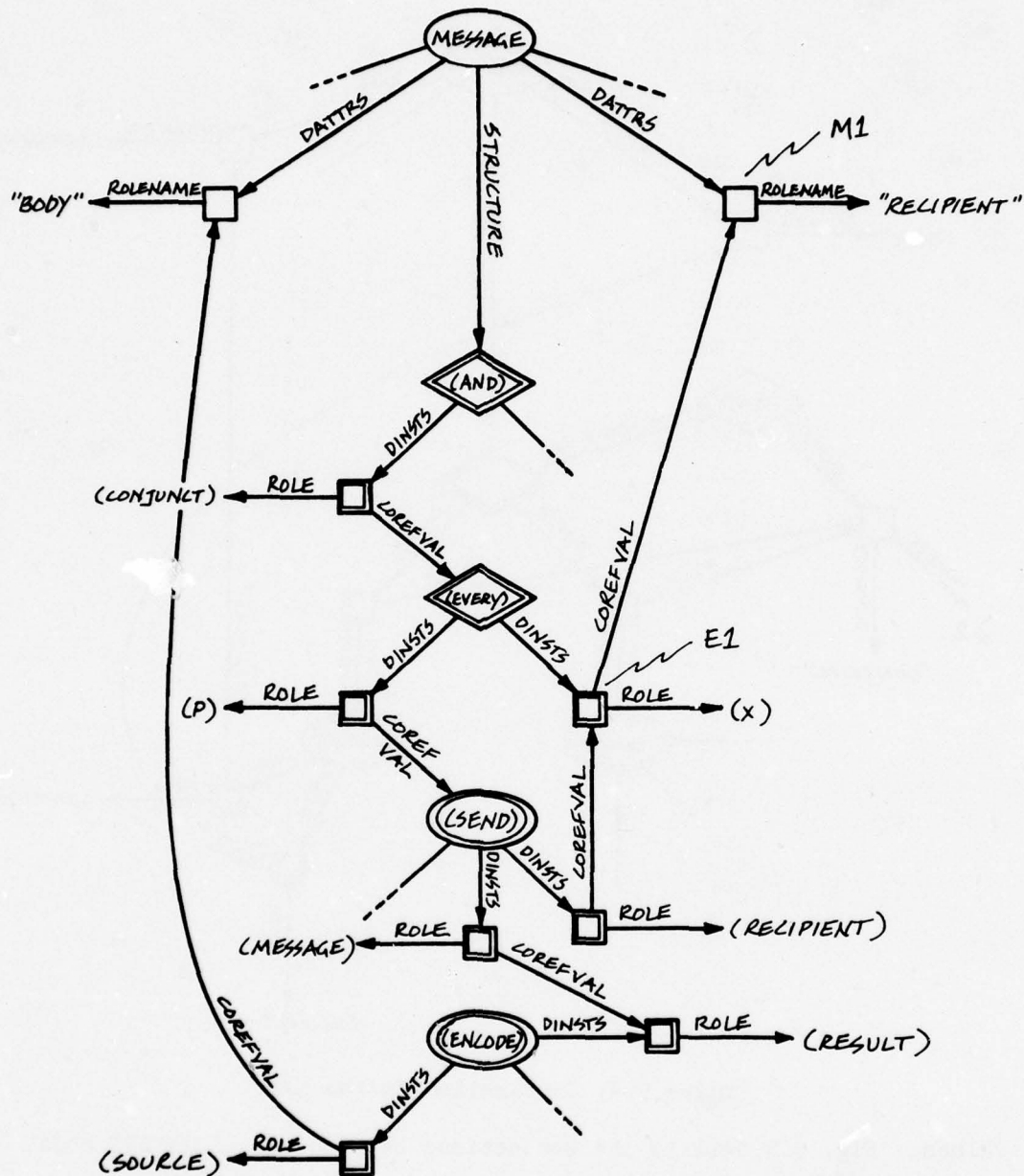


Figure 5.5. A universal quantification.

While our example of MESSAGE prescribes only a single filler for the BODY role, let us consider the structural condition if this were not the case: let us say, hypothetically, that MESSAGES can have more than one

BODY. With more than one text, we would have to quantify the SEND over the set of texts, in a way similar to the above treatment of recipients. But notice that the MESSAGE dattr of (SEND) does not point directly to the BODY role description node, but its COREFVAL is instead the RESULT of (ENCODE), which is a function of the BODY role. If we hooked up the SOURCE role of (ENCODE) to a variable quantified over the bodies of the MESSAGE, rather than directly to the BODY dattr of MESSAGE, we would have a structure expressing

FOR EVERY x / BODY ; SEND(SENDER, ENCODE(x), RECIPIENT)
(see Fig. 5.6). For each body, x, there would be some encoded version of x that would be sent by SENDER to RECIPIENT. Thus, we are really representing an existentially quantified variable (the encoded version) dependent upon the universally quantified one. This is the notation's equivalent of the formal logic "Skolem form" (see [Woods 1975a, p. 76]), in which existentially quantified variables are replaced by functions whose arguments are the universally quantified variables upon which they depend. In Fig. 5.6, (ENCODE) is a Skolem function, and its argument (i.e., its SOURCE role) is the universally quantified variable, x.

5.1.5. More complex accesses from S/C's

The "FOR" notation characterization of Fig. 5.6 was as follows:

FOR EVERY x / BODY ; SEND(SENDER, ENCODE(x), RECIPIENT) .

In Woods' [1968] procedural semantics for this notation, the class in such a statement (BODY, in this case) has a function that can successively retrieve its members. In the SI-Net graphical notation (Fig. 5.6), one can consider the COREFVAL pointer from node E1 to the BODY role description node of MESSAGE (node M1) an "access" or "focusing" function. That is, the dattr, when indicated in this way, focuses our attention on the desired subpart of MESSAGE (and when accessed from the x dattr of an EVERY quantification, stands for the enumeration of the set of values filling that role, one at a time). By

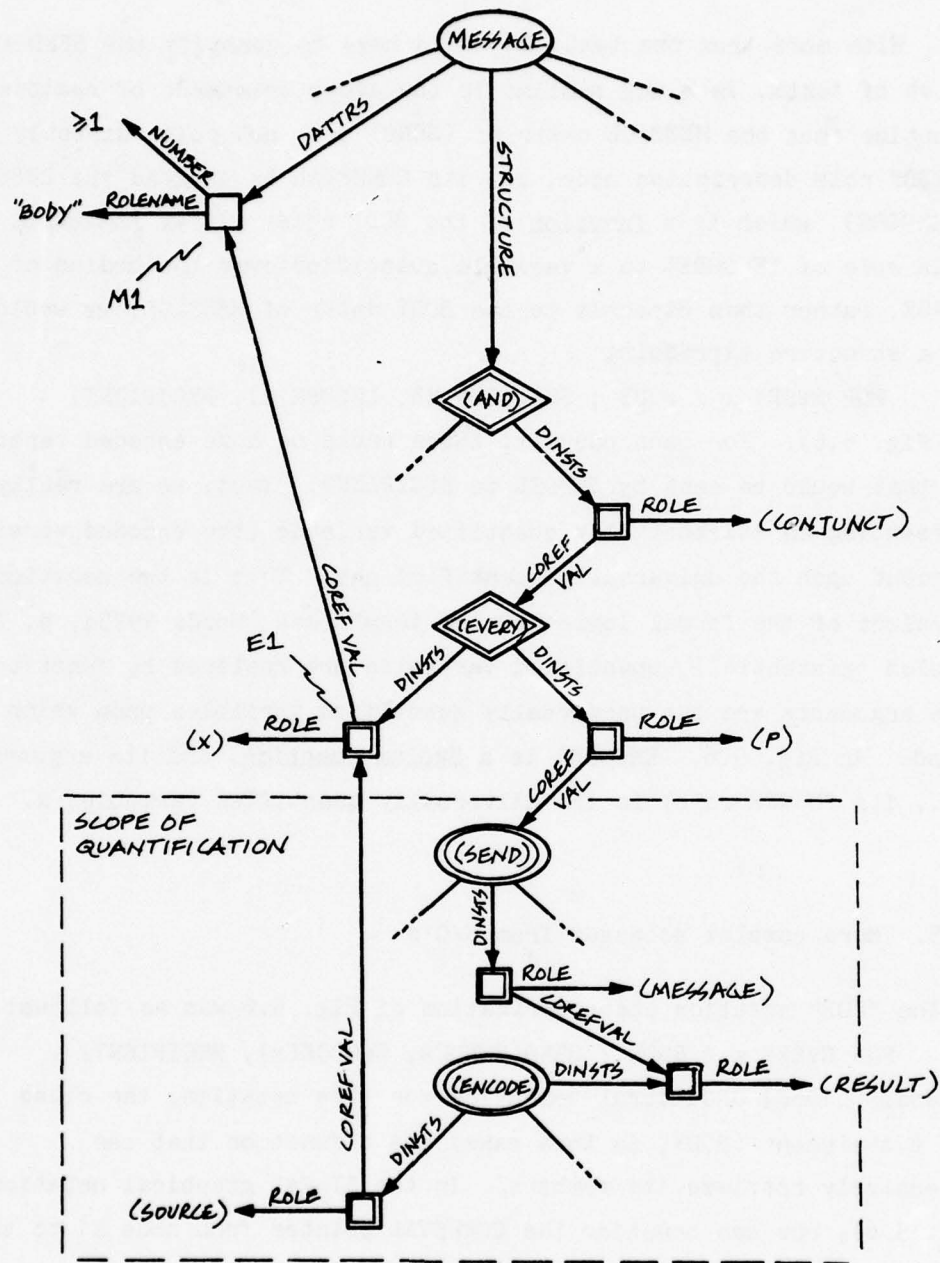


Figure 5.6. Skolemization.

the same token, we can consider the other two references to **dattr**s in the **FOR** expression as accessing functions -- **SENDER** returns the sender of the message when applied to a particular message, and **RECIPIENT** does

the same for its recipient. Notice that such a focusing function has an implicit argument -- the concept whose definition they are part of -- which defines the context for the access*. This reflects our general use of terms like sender and recipient; we say "the sender of the message", indicating a possible function-like interpretation.

SI-Nets allow us to make use of this interpretation of dattr to create composite access functions (e.g., "the name of the sender of a message"). A composition of accesses is represented by an independent role node (i.e., not connected with a DATTRS link to any concept node) which has a pointer to each functional part of the composition; the function to be applied first is indicated by a FOCUS link, while the second is indicated with a SUBFOCUS link.

A composite function representing "the name of the sender (of a message)" is represented in Fig. 5.7 by node F. When a COREFVAL link from the structural condition of MESSAGE points to this node, the functional composition indicates that first we focus on the SENDER of the MESSAGE, which is a PERSON, and then we focus on the NAME of the person (which is a STRING).

This compound access is necessary for the following reason. A COREFVAL pointer directly to node S of Fig. 5.7 would yield a PERSON who is the sender of the message. Assuming that we want the name of such a person, that single link is not sufficient to access the desired subpart of PERSON. On the other hand, a single COREFVAL link to node N would access the NAME of a PERSON; but notice that MESSAGE has two dattr whose VALUE/RESTRICTIONS are PERSON. Thus, the reference to the NAME

* The fact that we use dattr in this way allows us to make statements about entities in context. For example, in Fig. 5.1, the COREFVAL pointer to node H allows us to state something about hydrogen, in the context of its use in an H-bomb. A pointer directly to HYDROGEN would have asserted something about hydrogen in general, which is not the intent of such a definition. This is exactly what makes a dattr more than just a role.

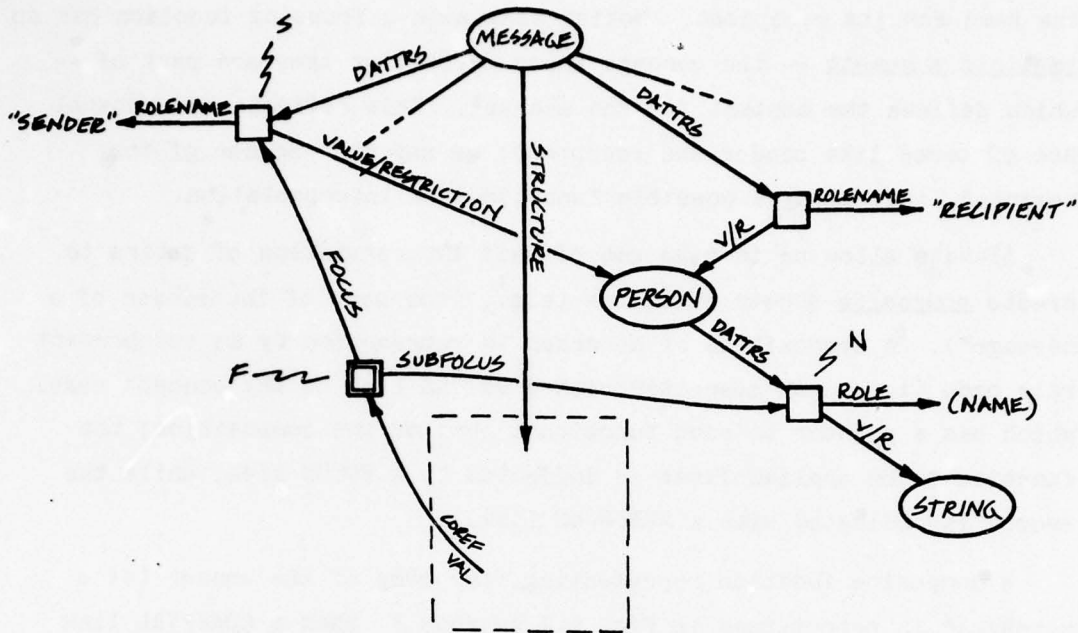


Figure 5.7. "The name of the sender (of a message)".

would be ambiguous between "the name of the sender" and "the name of the recipient". The compound function solves this problem by making the correct two-step access.

In addition, the composite dattr function can in turn access, as either FOCUS or SUBFOCUS, another composite function. This kind of recursive application provides arbitrarily long access chains where necessary.

5.1.6. A final note on structural conditions

With the structural condition, we have a powerful way to define new concepts in terms of those already known, primitive or otherwise. There is no need to insist on a fixed level of detail in concept definition to be taken as "primitive" in all descriptions. Complex concepts (like HYDROGEN/BOMB, for example) can be expressed in terms of a small number of other high-level concepts rather than as extremely complex

constellations of primitives. This type of representation facilitates the kind of idiosyncratic understanding that we expect from humans: while most people do not appreciate the full complexity of the workings of hydrogen bombs, they can express their own idiosyncratic understanding of the concepts involved in terms of those that they know well. Using the structural condition in the way that I have suggested, we can represent any of the different interpretations of a phrase like "hydrogen bomb" that we might encounter. Many representations prefer the "canonical" approach -- each concept has a single fixed, "correct" breakdown in terms of the conceptual primitives built into the notation (although other more idiosyncratic variations could, presumably, be represented)*. We can make use of high-level concepts defined in the same notation to reflect the common way of building new concepts on top of a foundation of previously learned ones.

5.2. Derived intensions

One of the benefits gained with the adding of a structural condition is a way to create new concepts in a semantic net; we can integrate new role descriptions (dattrs) through a structural condition built from previously learned concepts. Another important source of new concepts in the network framework is the derivation of "subconcepts" from the descriptions of very general concepts. Just as the representational duty of a concept node is seen to be more than just a placeholder for a class, a subconcept node is to express more than just an extensional subset. A node for some subconcept X' of X is usually thought to express the definition of X' as "an X such that . . .", where ". . ." is some restriction of the definition of X. Unfortunately, the way to construct a node that expresses this "such that . . ." qualification is

* See [Woods 1975a] for a detailed criticism of canonical approaches.

not always clear.

The lack of a well-defined subconcept mechanism in most semantic net notations appears to be similar to the confusion we saw above with assertions about "instances" (individuators, to be precise). That is, as Woods points out [1975a, p. 59], a node that expresses a modification of a general concept in the manner of

N12368
 SUPERC TELEPHONE
 MOD BLACK

is ambiguous in intent. The "MOD" link may carry the assertional import that DINSTS links have (and thus make the node mean "all [or some, depending on implicit quantification] telephones are black"), or it may be definitional as DATTRS links are (and thus the node would represent the concept of a BLACK TELEPHONE). In addition, the node could also be taken to represent telephone N12368, a particular telephone which is black. This last case is the real "assertional" case, and it would be represented with INDIVIDUATES instead of SUPERC and DINSTS to a role node instead of MOD.

In order to make the representation of subconcepts well-defined, we need to specify a modification mechanism that clearly differentiates between intensional modifications and incidental insertions of particular values. Here we investigate in depth the definitional mechanisms introduced informally in Section 4.1.2. Subconcepts are formed by manipulating the two main structural aspects of concepts -- the definitions of the parts (the dattrs), and the structuring condition. Herein lies the "structured inheritance" of these networks.

5.2.1. Role-oriented modification

First, we shall look at how to derive subconcepts by manipulation of an individual dattr. There are several ways that role descriptions can pass information to lower nodes. Here I discuss in more detail the three that I mentioned in Section 4.1.2 -- restriction, differentiation, and particularization.

5.2.1.1. Restriction

One type of modification we might make to create a new concept is the restriction of a dattr. For instance, we might be able to characterize the FUNCTION of a particular class of commands as "PRINTING the CONTENTS of MESSAGES" (in appropriate network notation), and we may wish to create a node for a subclass of such printing commands whose function is "SUMMARIZING the CONTENTS of MESSAGES", where SUMMARIZING is a subconcept of PRINTING. Such a modification is accomplished by 1) creating a node to represent the subconcept, 2) creating a node to be associated with the subconcept that will correspond to the role description node of the parent concept (and placing the appropriate modifications at that node), 3) linking the new role modification node to the subconcept node by a primitive link indicating intensional modification (DMODS), and 4) associating the modification with its appropriate functional role by linking the modificational role node to the role description node of the parent; this is illustrated in Fig. 5.8.

Individuators can be associated with such a subconcept in the same way as they could with the undifferentiated concepts that I discussed in Chapter 4. Modificational role nodes inherit all information from the role description nodes that they modify, except for the link that is explicitly overridden (VALUE/RESTRICTION in Fig. 5.8), and act just as role description nodes do -- they constitute descriptions of parts of

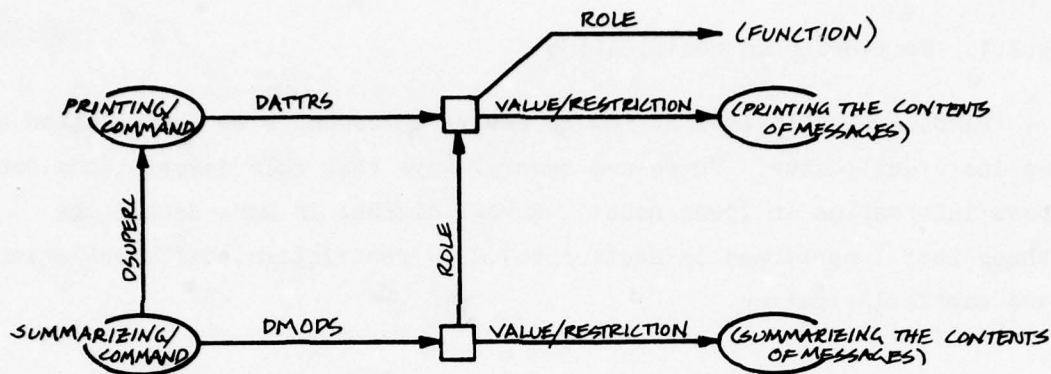


Figure 5.8. A subconcept.

potential instances, and can be further modified or instantiated.

The type of linkage introduced here makes explicit the intensional connection between a concept and one of its subconcepts. The part of the concept being restricted is explicitly indicated by the ROLE link, rather than being implicitly referred to by a repetition of a conceptual relation name. Such a detailed linkage, while more difficult to read than a simple name repetition, avoids confusion and potential ambiguity.

5.2.1.2. Role differentiation

Another interesting modification that one can make to a general concept is the further specialization of a kind of part. For instance, a node for COMMAND would indicate that commands take ARGUMENTS (see Fig. 4.15). For commands in general, it may not be necessary (or possible) to specify the number and type of the fillers of this role. We would like, however, to be able to pin down for a particular command group (some subset of COMMAND, such as PRINTING/COMMANDS) the exact number and type of each argument to expect. For a given role, we may wish to express subroles that are subsumed by that role. That is, we would like a way to differentiate the role of ARGUMENT. In the spirit of the above explicit representation, we can create new dattrs for the subconcept and relate them to the parent by primitive links, as illustrated in Fig.

5.9. The nodes indicated by the DIFFS links are role description nodes,

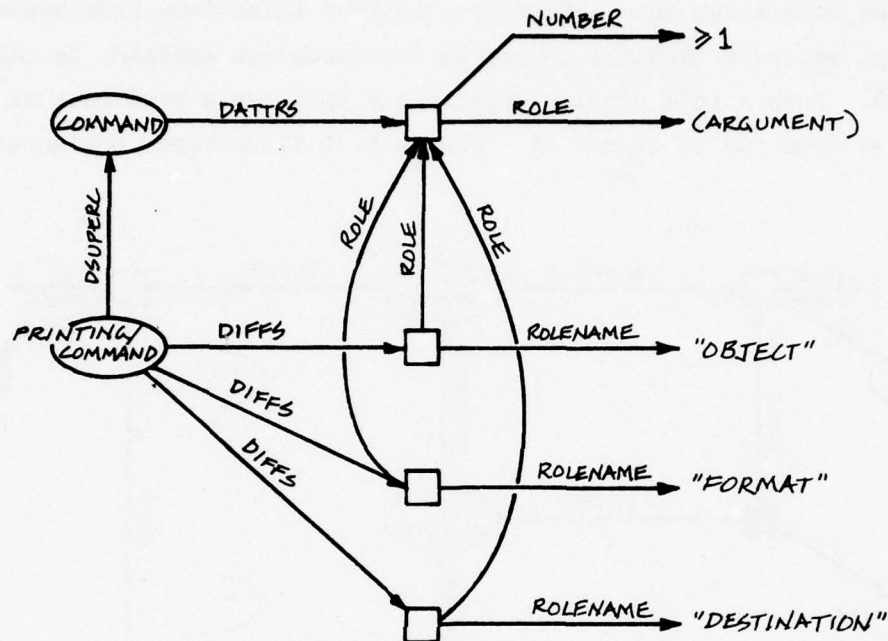


Figure 5.9. Differentiated roles.

and provide the intensional basis for individuators to be derived from the PRINTING/COMMAND concept (see Section 4.3.3). As a result, they, too, have roles associated. However, their meaning in part depends on the dattr descriptions that they differentiate.

5.2.1.3. Particularization

In some cases, a subconcept may fix for all further subnodes one of the fillers of a role required by its parent. That is, a part of the description may be instantiated as part of the definition of the new concept (rather than as an assertion about the members of the class). This particularization (binding) can be accomplished explicitly in a manner similar to those modifications presented above: the attribute/value pair is indicated by a special (role instance) node,

which is connected to the dattr (role description) of the parent concept and to the subconcept node itself by primitive links (the link between subconcept and role instance nodes, as I pointed out earlier, is called "DINSTS"). Such a role binding essentially produces a predicate of degree $n-1$ from one of degree n^* . Figure 5.10 illustrates how we might

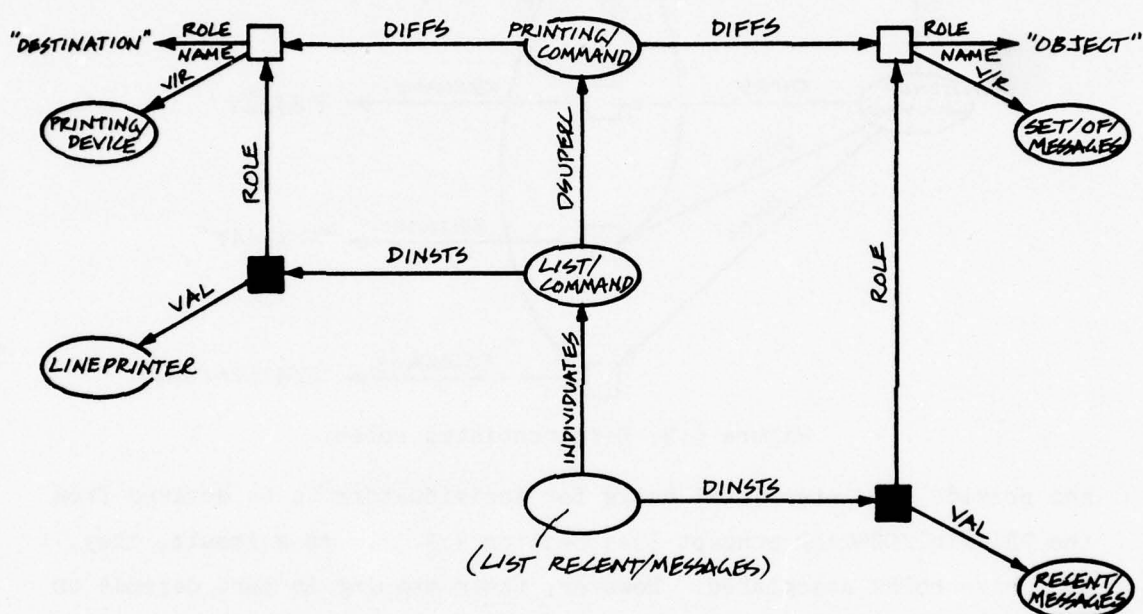


Figure 5.10. A particularized concept.

further specify the PRINTING/COMMAND concept of Fig. 5.9 (here I use only two of the role descriptions created in that figure). The node for LIST/COMMAND fixes the destination for all concepts to be found below it by particularizing the DESTINATION role with the value, LINEPRINTER. Since the DESTINATION of a LIST/COMMAND is thereby always specified, the

* Particularizations (and some restrictions) are only permitted when the structural condition indicates that a change to a dattr can be made independent of all other dattrs (for example, a relation like BETWEEN might require the two extremity dattrs to be instantiated simultaneously in order that a particularization or relative, spatial restriction be well-defined). Dattrs are not necessarily independent.

node representing "LIST RECENT/MESSAGES" needs only account for the OBJECT role to fully individuate the defining concept. (Note that, while not explicitly indicated in the figure, LINEPRINTER is an individuator of PRINTING/DEVICE and RECENT/MESSAGES is an individuator of SET/OF/MESSAGES.)

5.2.2. More global modifications

I will now discuss briefly two other important ways to derive concepts from existing concepts. These types of derivation deal more with the concept as a whole than with its isolated parts.

5.2.2.1. Modification of structural condition

In addition to altering the description of an individual part, we might want to derive a new concept by redefining how already specified parts interact. For example, consider the derivation of the concept DIABETIC from the concept HUMAN/BEING. All of the parts are the same, but the way that they work together is slightly different. An important relationship between these two concepts would be lost if we were forced to define DIABETIC using only the facilities mentioned above (we would somehow have to redefine all of the affected parts, and indicate their new interrelation in its entirety).

While the modifications that I spoke of above (Section 5.2.1) dealt only with the individual dattr's, the obvious place for an explanation of a disorder like diabetes is in the functional interrelation between those roles -- that is, the structural condition. A systemic dysfunction could thus be represented by including a structural condition addendum in the new concept, and having that inherit the structural condition of the parent concept, with the ability to override relevant parts. That way, only the appropriate parts of the structural

condition may be altered, without having to resort to a complete redefinition*. This effect may be achieved with a single link type, "PREEMPTS", between paraindividuals in the S/C's of the parent and the descendant.

5.2.2.2. Analogy

A very important kind of structure-specifying operation for a knowledge-acquiring system is analogy**. Suppose, for instance, that we were to create a node for AUTOMOBILE, a subconcept of MOTOR/VEHICLE. While MOTOR/VEHICLE might specify only a requirement of "at least two" WHEELS, AUTOMOBILE would differentiate this requirement into two steerable FRONT/WHEELS and two powered REAR/WHEELS. Now, consider a natural definition for the concept of an automobile with front-wheel drive. We most likely would not expect to redefine the entire concept of AUTOMOBILE, substituting only FRONT/WHEELS for REAR/WHEELS in the explanation of the automobile's powering. Instead we would prefer to

* We might alternatively merely add an addendum to the structuring condition that indicated that the subject had to take insulin, if the level of description called for a non-physiological explanation. In this case, the added section of structural condition would be considered to be conjoined with the parent one, without overriding any of the previous structural information.

** The general ability to view something as something else ("Can a shoe be a hammer?") is a critical one for any system that attempts to use its current store of knowledge to interpret a new input. I have dealt with this at length in an earlier paper [1975] and will treat it in Section 6.5.1, and it is one of the motivating factors behind the KRL language [Bobrow & Winograd 1977]. Recently, I have discovered (with the help of Rusty Bobrow) that the structural condition can provide a great deal of guidance when trying out an analogy. Once a role-value binding is made, the structural condition can be looked at for relationships that might exist between parts of the target structure, in a way that could lead to the making of further bindings. One important kind of analogy asks if the parts of one thing (e.g., a shoe) can be mapped onto the roles of another (e.g., a hammer).

say, "just like AUTOMOBILE except powering applies to the front wheels, not the rear".

A simple way to express such an explicit analogy-plus-modification is to use a link-type called "DBROTHERC", between the target node (e.g., AUTOMOBILE) and the newly-defined node, and have only the modified features appear at the new node. This link would be interpreted as passing all role definitions and the structural condition from the target node to the new one, except where explicitly modified (the powering explanation, for example, could be expressed at a role description node -- as a value restriction concept for "powered wheel" -- or in the S/C -- indicating that power is transmitted from the engine through the drive train to the rear wheels). This is the same interpretation given to DSUPERC, except that nodes connected by DSUPERC are partially coextensive, while those connected by DBROTHERC usually reflect mutually exclusive classes. DBROTHERC could be considered a shorthand which reflects the natural explanation, and which avoids the repetition of a complex structure at a node that has virtually the same connection to its parent as its brother node. In addition, DBROTHERC allows the two subconcepts of the same parent to remain "in sync" -- any modification to the one pointed to by DBROTHERC will be automatically inherited by the one pointed from.

5.3. Consequences of intensional structure

Without the mechanism of intension in our repertoire, we would have had a hard time explaining precisely the import of many of our network links. In fact, as we saw, most often standard links attempted to embody conceptual relations rather than epistemological ones; since the meaning of a concept is embodied in all things that can be accessed from the concept, these conceptual links would have complex semantics that would vary depending on the moment-to-moment content of the data base.

As a result, they are not good candidates for representation primitives. In addition, in the older notation, mixed in with conceptual relations like COLOR, LINTTEL, etc., there were relationships like ISA, INSTANCE/OF, etc., which had virtually unexplainable import.

Now that I have motivated intension as a way of interpreting network nodes, we can better understand some of the formerly mysterious characteristics of semantic nets. The links in SI-Nets will be consistently defined at the same level, and thus a link like INDIVIDUATES can be defined in terms of its effect on nodes which are tied together with DATTRS, DINSTS, DMODS, etc., links. This final section looks at some of these now more easily understood representational phenomena.

5.3.1. Passing structure -- the DSUPERC link

The modification mechanism makes use of a primitive link type to express the concept-subconcept relation (DSUPERC, for "define as superconcept"). While we see links similar to this in many networks, it is rarely clear what their total import is. We here see that the purpose of the DSUPERC link is to pass intensional structure, as if it were a cable. At first glance this appears to mean only that all properties of a node at the head of a DSUPERC link are assumed to apply to the node at the tail of the link -- this "inheritance of properties" is one of the standard benefits of network notations. However, there is additional meaning in the DSUPERC link -- the internal structure of the offspring node is defined by the dattrs and structural condition of the parent. That is, the set of functional roles applicable to the parent concept is passed on to the subconcept. Thus, any modification or instantiation is made within a precisely defined context, and the functional role of the restricted dattr within the complex is carried through to the subconcept or individuator. This is essentially the case in other networks, except that, as we have seen, their role links are

usually used ambiguously and their semantics is rarely precisely specified.

The DSUPERC link essentially imposes the case frame structure of a concept onto a subconcept. That is, it provides a structured inheritance. This determines the kind of links that one expects to see at the subconcept node, and gives meaning to the "cases" found there.

5.3.2. Individuators as individual concepts

Much of the above discussion about subconcepts is reminiscent of the earlier treatment of individuators (Section 4.3.3). The structure of the constellation of the DINSTS links (and the meaning of the instantiated roles themselves) is defined by the parent concept node's dattrs and structural condition. In light of this similarity, we may interpret the node for an instance as representing not just the individual referred to, but an intensional description of that individual. This is what distinguishes an "indivuator" (a description) from an "instance" (a thing in the world). For example, the node for John tells us that John is the person who . . . (where ". . ." represents characteristics criterial to being John). This interpretation follows closely Carnap's definition of an individual concept, which is the intension of what he calls an "individual expression". Two types of entities constitute individual expressions: 1) a designator of the form "the x such that P(x)", where P is a predicator; and 2) the full expression of a functor, for example, "3+4". An instantiated functor designates a single individual, the value of the corresponding function. Thus "3+4" and "(4*2)-1" would designate the same individual, 7. But notice that the two functors express the description of the individual in two different ways. The instantiated functors would correspond directly to SI-Net individuators, with the values being captured by the RESULT or WHOLE dattrs. The indivuator represents how the value of the function (parent concept) is to be

derived from the role fillers, and the RESULT dattr specifies the particular derived value. The INDIVIDUATES relationship implies that the individuating concept purports to represent one and only one individual in the world*.

An example will help make this clearer. Consider the function, DISTANCE(x,y), which returns the value which is "the distance" between points x and y. We have two language expressions (designators) which are used at different times to refer to the distance between two places, say Boston and Philadelphia: one designates the value of the distance -- "325 miles" -- while the other designates the concept of the distance -- "the distance between Boston and Philadelphia". (The latter is what Carnap calls an "individual expression".) In the first sense, the distance between Boston and Philadelphia is the same as the distance between any other two places which are 325 miles apart; in the second, it is the same as no other distance.

In SI-Nets, we differentiate between the two types of expression (the two senses of "the distance") by interpreting the individuator node representing "the distance between Boston and Philadelphia" as a representation of the intension of that individual expression (as I have mentioned, Carnap calls this intension an "individual concept"). The representation of the value of the function application (the thing that "325 miles" refers to) then follows as the thing pointed to from the RESULT role instance node of that individuator (see Fig. 5.11).

In Fig. 5.11, node B-P represents "the distance between Boston and Philadelphia", while node D represents the value itself. Nodes B-P and F-M are distinct, and each's individual constellation of links

* Note that interpreting individutors as individual concepts (i.e., intensional descriptions of individuals) implies that the same individual might be represented by more than one individuator. This is not usually acknowledged to be the case in standard networks, although it is embodied in the "perspectives" of KRL [Bobrow & Winograd 1977], and can be handled by Hendrix's "e" (as opposed to "de") links [1975a].

Section 5.3.2
Individuators as individual concepts

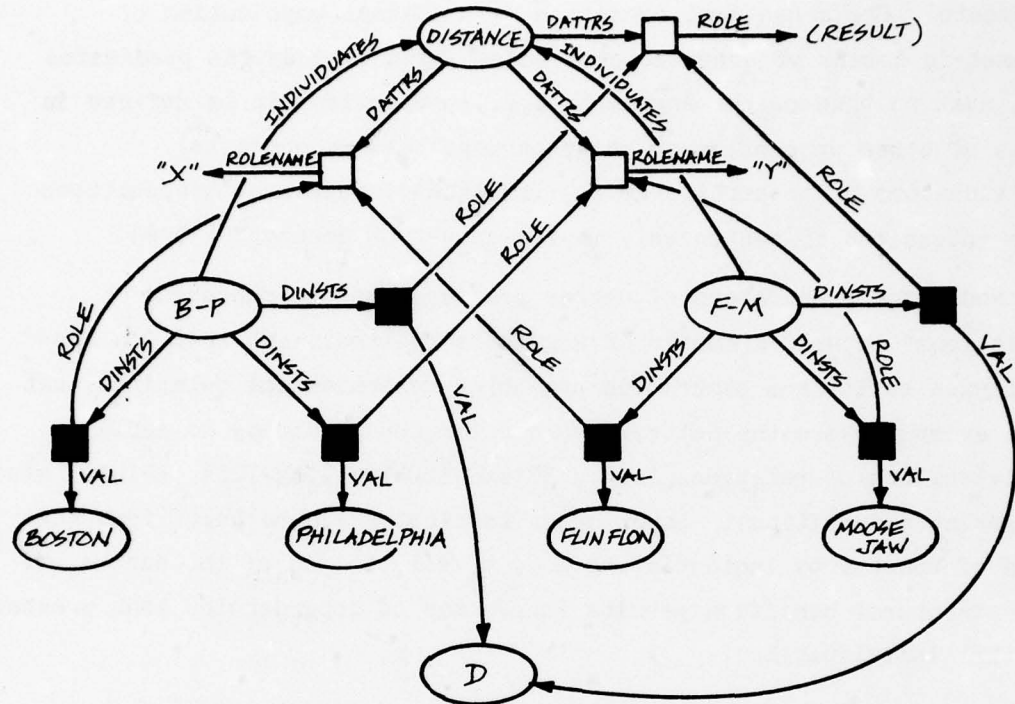


Figure 5.11. Individual concepts.

represents its intensional structure. They are intensionally similar, but not equivalent. However, in the world being represented, they are extensionally equivalent, since their RESULTS are the same concept. Notice that node D is called "325 miles" only in one particular value system; it could just as well be "523 kilometers", etc.

5.3.3. Using intensional structure to understand networks

As just illustrated, an appreciation of the intensional nature of concepts in a semantic network can help us to understand the fundamental nature of the formalism. We can use this view to understand the subtle differences between the various kinds of concepts normally represented in networks. Nodes for predicates of degree one represent "properties", and have a single dattr representing the argument to the

predicate. The structural condition is a logical combination of parametric tokens of other concept nodes which express the predicates that must be true of the argument (i.e., properties can be defined in terms of other properties or relationships between concepts). Individuators of properties (e.g., RED(ARCH1)) represent propositions (the intensions of sentences), in the manner of Schubert [1976].

Nodes for predicates of degree greater than one represent "relations" between a number of arguments (dattrs). Their S/C's are complexes built from other concepts which determine the relations that must exist between the dattrs in order for the predicate to apply. Individuators of relations (e.g., BETWEEN(BOSTON, NEW/YORK, PHILA)) also represent propositions. Lower order predicates can be built from this kind of concept by instantiating some or all but one of the dattrs, if the structural condition permits (there may be dependencies that prevent partial individuation).

Section 5.3.2 illustrated what nodes for individual expressions mean. Finally, nodes for functors and objects represent "functions" of some number of arguments. The critical difference between this kind of node and a "relation" is that individuators represent individuals rather than propositions (recall the definition of an individual expression). The RESULT and WHOLE dattrs capture the individuals as entities, while the individuator nodes and their links represent the intensional structures describing the individuals. This is one of the central operations of the semantic network notation -- the representation of individual structured objects -- yet its place is not clear until we appreciate the intensional nature of the representation.

Chapter 6. Understanding Nominal Compounds

The structured inheritance paradigm for representing knowledge that I have presented differs significantly from most previous semantic network formalisms. Links in the network represent only primitive "epistemological" operations for building and relating concepts, and never stand for any conceptual or case relationships. Thus the intuitions that one uses to build from his representational repertoire a particular data base will not carry over from the old semantic network lifestyle. While this is not necessarily a detriment -- for as we have seen, the easy way in which one might build semantic nets leads to inconsistent, ambiguous, or incomplete structures -- we still need to develop a feel for the encoding of a domain in terms of dattrs and structural conditions.

To this end, this and the following chapter will illustrate how the structural paradigm might be applied to real-world knowledge. The examples presented here will serve to exemplify a methodology for building the representation of the concepts of a particular domain out of sets of roles and their structural interrelations. I hope to provide a feel for how one should go about implementing concepts using the intermediate-level notation of DATTRS, VALUE/RESTRICTION, DINSTS, etc. Such guidelines for building nodes given a set of node building blocks, while missing in most semantic net presentations*, are very important to the efficacy of such a general representation scheme as SI-Nets.

* Bell and Quillian's "Capturing Concepts in a Semantic Net" [1971] is the only attempt (known by me) at a primer for building concepts out of representation pieces, and unfortunately, their formalism is inadequate.

Besides illustrating how one might use the structural representation, these two chapters will serve to show that the domains of nominal compounding and the Hermes message-processing program [Myer, Mooers & Stevens 1977] can be perspicuously modelled in a machine-interpretable formalism. Compound nominals have not heretofore been analyzed in a way that would allow a computer to make intelligent use of the mechanism in understanding English text (annotated bibliographies, in particular). No program exists that can understand never before seen nominal compounds -- that is, that can use knowledge of conceptual relationships to determine the underlying associations between elements of a new compound (but see [Rhyne 1975] for a computational account of the process of generating compounds). Moreover, the Artificial Intelligence world has yet to see an intelligent assistant program that might accept new knowledge and use it to assimilate further new information, or to interpret and answer requests couched in terms different from, but conceptually related to, those in which the original description was offered.

Offered here are not complete solutions to these very deep and difficult problems, but substantial beginnings of solutions and an accompanying methodology. As I stated in Chapter 3, it is that methodology (of which the Structured Inheritance Net is only one possible result) which is the keystone to representing sophisticated and highly interconnected knowledge domains. The practice of making every relation uniformly a link breaks down when we encounter a sphere of knowledge with the complex and structured intensional relationships of even a simple message system like Hermes. Only when our basic approach allows us to sort out the different types of fundamental knowledge units and the connections between them do we stand a chance of constructing really usable, accurate representations.

Chapter 6 will attempt to convince you that the structural paradigm we have developed is a reasonable way to attack the understanding of nominalizations and nominal compounds. This chapter will highlight the

similarity between nominals and verbals in SI-Nets, and will show how the representation of idiosyncratic interpretations is central to the compounding domain and can be reasonably approximated by an SI-Net representation. Chapter 7 will follow with an attempt to show how a data base of knowledge about Hermes might be constructed. I will touch on all aspects of the program that are necessary to ensure the intelligence and helpfulness of an on-line Hermes consulting program. Such a consultant would require an extensive knowledge of all procedural as well as static features of Hermes. This chapter will detail the internal structure of objects, individuation, and the nature of inheritance within the paradigm. In addition, it will help to point out the important place of intensional definition in domains such as Hermes, and how SI-Net formalisms are well-suited for heavy reliance on intensional operations.

Not only will these two chapters illustrate how to use the structures of Chapters 4 and 5 and how to represent knowledge in our two domains, they will also show how the new network scheme holds up under the stress of some difficult representational problems. It is interesting to remain aware, as we get involved in these two areas of knowledge, of how disparate the domains are, yet how amenable they both are to representation in terms of dattrs and structural conditions.

6.1. Understanding English nominal compounds

One area in which a general, extensible memory representation for human knowledge might serve as an extremely useful tool is that of the organization of textual information. For example, an ever-expanding personal library of documents and notes might be kept under conceptual control by a program that could accept as inputs descriptions of the textual sources, and integrate these comments with the descriptions of all previous source materials and some general "knowledge of the world".

If this integration were based on the conceptual content of the annotations, rather than merely a surface look at the words used to express that underlying meaning, then the program could hope to perform intelligent operations such as building reading lists based on complex conceptual similarities between references and queries rather than on an artificially limited, predetermined set of descriptor terms.

One well-known form of information-compaction device that might be suitable for a first attempt at a library assistant program is the annotated bibliography. Annotations are used to abbreviate (and editorialize) the content of much larger textual sources, yet they generally make use of the same range of concepts and language constructions as the documents themselves. Thus, while a good source of compact descriptions, bibliography annotations make demands on an understanding program as severe as those made by general texts. To handle the concepts introduced in brief abstractions of more extensive texts, we need a representation general enough to handle the important concepts of the texts themselves.

Let us look at some examples to make the discussion more concrete. In a bibliography discussed in detail in [Brachman 1973], we find annotation constructions ranging from "Reasonably understandable," to "BBN semantic nets and the inference problem," to "Implementation details of a parsing system for ATN grammars," to "Discusses some of the problems involved with this formalism," to "This book consists of papers delivered at a New York conference in 1971." Thus, any program that might take these annotations as input must be prepared to handle adjective, noun, and verb phrases, as well as complete sentences, and conjunction and anaphoric reference. Moreover, specific concepts discussed in the text of the references themselves (such as ATN grammars, semantic nets, etc.) must be understood to some reasonable extent before these notes can be stored in the appropriate way.

For example, take the phrase, "Implementation details of a parsing system for ATN grammars". Some information about what a parsing system

does is necessary to understand how a "grammar" relates to it (the relation is indicated in the phrase by only a non-specific "for"). Further, the differences between a system, in the sense of computer program, as intended above, and a more formal notion of system, must be appreciated in order to see where "implementation" fits into the phrase (here indicated only by "of"). This constant lack of detailed information on how to conceptually tie together the content words is one of the dominant characteristics of annotations, and dictates that a powerful conceptual representation is needed to draw together, inferentially, the important pieces of a description -- pieces which are most often connected only by non-contentful devices, such as prepositions and juxtaposition (as in compounds; see below).

Finally, notice how much of these phrases constitutes "syntactic sugaring" -- concepts like "details", "some of", "problems", "book", and "papers" do not add to our descriptions of the topics of the references, even though they do provide descriptive details that might be of use once we have located the semantically-designated set of references that we want. A close look at bibliographic annotations reveals that a good deal of their expressive effort is devoted to these terms that do very little to help us distinguish between the topics of the references. For purposes of determining what it might take to relate documents according to what they are about, we will ignore these "non-topic-specific" terms (see [Brachman 1973] for thoughts on a grammar for such constructions). Rather, we will focus on how to specify the topics of documents, and see if a representation can be devised to relate topic specifications in a conceptual way. It is in the representing of an open-ended domain such as the topics of documents that the power and the foibles of a knowledge representation scheme will become clear.

Notice that the document topics are invariably expressed by noun phrases or nominal compounds that can be transformed by periphrasis into noun phrases (usually modified by relative clauses, e.g., "parsing system" to "system which is used for parsing [sentences]"). What, then,

would it take to represent accurately the underlying conceptualizations of these critical keys to bibliography indexing? The crucial problem in achieving an intelligent document assistant lies in finding the relations between all of the nominal pieces of a topic description, and building up a memory structure that accurately reflects the complex meaning of the phrase or compound. At first blush it seems easy to determine what a compound like "ATN grammars" must mean -- we do it so quickly and so often that compound generation and understanding are second nature. Yet nowhere in the phrase itself are there any clues (except word order) that tell us what structure to build from the words or what inferences to draw from the resultant complex. It may be easy to find the referents of "ATN" and "grammar", but there are a myriad of potential ways to make connections between those two concepts*.

Thus a significant problem of compounding is the isolation of the single intended connection from a set of many reasonable alternatives. Yet before we can even consider the choosing of the "right" underlying structure, we must be able to determine just what the alternatives are. As the Gleitmans insist, this fundamental problem is not so easy as it first appeared: ". . . while it is easy enough to transform a two-noun compound into a relative clause, the problem of finding the appropriate linking bond is often far from negligible, for the bond must be both plausible and intimate." [Gleitman & Gleitman 1970, p.178]

* As Gleitman and Gleitman [1970, p. 90] point out, for example, there are at least five reasonable ways to connect the concepts HORSE and CART in the compound horse cart: "Very different expressions are related to the same compound. In principle, horse-cart may mean cart that is shaped like a horse (as box-car means car that is shaped like a box), and similarly it might mean:

- cart that is drawn by a horse (as in dog-sled)
- cart that a horse rides in (as in passenger-car)
- cart for a horse (as in hay-wagon)
- cart that is as big as a horse (as in horse-radish)."

The usual intended meaning is established by use.

In this chapter, I discuss what the shape of "intimate" connections between concepts might be, and how the network formalism developed in Chapters 4 and 5 provides a reasonable mechanism for representing the meanings of nominal compounds. It should be emphasized that the solution to this problem lies with the type of conceptual information that the SI-Net formalism handles, rather than with syntactic considerations, since the only syntactic cue that exists between parts of a compound is word order*. Thus it is up to the conceptual representation of those parts to offer candidates for associations between them**. The imposition on concepts of the epistemology embodied in dattr, etc., gives a strong push from one concept to others intimately (or potentially intimately) connected to it in just this way. Here I examine in some detail how the epistemology provides a useful tool for representing and manipulating nominals and the connections between them. As I have mentioned, this is the key to representing document topics so that annotations might be read, indexed, and ultimately retrieved in an intelligent manner.

Two final notes before I proceed to an in-depth look at the representation of compounds -- first, one might propose that we represent all compounds that we expect to encounter in a bibliographic corpus in advance, as lexical units with predetermined structure. However, as has been pointed out in many places, compounding is one of the most productive mechanisms in English***. We can find a reasonable

* "... We have thus implicitly relegated the problem of the appropriate verb to the semantic component of the grammar." [Gleitman & Gleitman 1970, p. 97]

** Marchand [1966, p. 22, quoted in Gleitman & Gleitman 1970, p. 92] states, "... 'In forming compounds we are not guided by logic but by associations. We see or want to establish a connection between two ideas, choosing the shortest possible way. . . .'"

*** Gleitman and Gleitman [1970, p. 65] for example, state that "They [compounds] are a relevant constructional type, for apparently they can be derived only by reference to the kind of recursive processes that

interpretation for virtually any pair of adjoined nominals. Consider how many compounds have been created in just this section so far: "memory representation", "reading lists", "descriptor terms", "library assistant program", "language constructions", "bibliography annotations", "knowledge representation scheme", "inference problem", "New York conference", "document topics", "document assistant", and "topic description", to name just a few. It should be clear that a general mechanism is necessary for the processing of such compounds since it would be impossible to determine in advance the range of combinations. A mechanism with power like that of the Structured Inheritance Network notation is a necessity, rather than a luxury.

Second, in order to represent accurately the meanings of compounds, the mechanism must be prepared to deal with a certain kind of idiosyncratic variation. Each person has an interpretation of a compound that is tailored to his own conceptual repertoire, and thereby different from that of another person's. Even well-known and conventionally specified compounds can be expressed in different ways, at varying levels of detail. The example of Section 3.3.1 regarding "lion house" is one case in point. Another comes from Lees [1963, p. 123]:

. . . consider the compound pontoon bridge. In this case it is not even obvious which interpretation is the most commonly used, but the following ones might occur to us:

bridge supported by pontoons	(like <u>steamboat</u> = boat powered by steam)
bridge floating on pontoons	(like <u>seaplane</u> = plane landing on the sea)
bridge made of pontoons	(like <u>blockhouse</u> = house made of blocks)

provide the basis for novel syntactic behavior. . . . Further, they are a central combinatorial device in English. The creation of complex compounds is a frequent and familiar productive activity, one which shows up at a relatively early stage of development, and one that is used without restraint even in the most rudimentary discourse."

pontoons in the form of a bridge (like cell block = cells
in a block)

Thus any representation that attempts to express the interpretation of these kinds of compounds must account for idiosyncratic descriptions -- that is, descriptions in terms of the particular set of concepts available at the time of definition. What this means is that we cannot expect to handle this task with a set of "canonical" definitions; the representation must be flexible enough to express many different interpretations. Some of these definitions may be far from "complete" or "correct". As determined in Section 5.1.3, our SI-Net representation affords just this type of expressive power. We shall see in examples of each of the compound types to be presented that the definitions given are only single members of sets of many variant interpretations.

6.2. The Grammar of English Nominalizations

A comprehensive attempt at understanding the mechanisms of nominalization and compounding was made by Robert Lees in his 1963 book, The Grammar of English Nominalizations. In this classic effort, Lees derived one of the earliest transformational grammars and illustrated how various kinds of nominalization might be transformationally derived from a base component. Lees dealt with both the sentences from which nominals could be produced ("constituent sentences") and those into which the derived constituents could be inserted ("matrix sentences"). While his accounts of the functions of noun phrases and the derivation (from verbs) of certain nominal expressions are meticulous and impressive, Lees' major contribution is his comprehensive enumeration of many distinguishable types of compounds that one finds in English.

6.2.1. A sketch of Lees' account

Since, as Lees states, "it happens that in all expressions of this sort the first constituent is attributive to the second" [1963, p.116], he first tries to account for compounding by simply preposing a post-nominal modifier (a predicate NP), which in turn is derived through a relative clause transformation:

"the course is a snap---> course which is a snap--->
course a snap---> snap course" [p. 116]

This, however, does not account for the multitude of compounds which have "no source sentences for this kind of adjectival derivation" [p. 116]. For example, car thief would have to be derived from an ungrammatical "The thief is a car." This leads Lees to postulate that it is not only predicate nouns that can be preposed, but objects as well, thereby allowing car thief to be derived from "the thief steals the car."

Yet this, Lees concludes, is still not sufficient to explain that while windmill and flour mill express the same kind of subject-object relation, their order is reversed (i.e., "Wind powers the mill," but "The mill grinds flour"). Given that other interpretations can be found that make these compounds identical in underlying structure (i.e., explosive flour dust could be used to power the mill, and huge wind-generators used in wind tunnels could be considered to be "wind-mills"), Lees proposes that we might get by with the suggestion that such compounds are to be derived from noun-verb-noun sentences. To back this up, Lees states that "given any two English (concrete) nouns N1 and N2, it seems always to be possible to find sentences of the form N1 V N2, as well as of the form N2 V N1, for some V's." [p. 117]

Alas, there are still many compounds which do not subscribe to these rules. Lees lists an impressive array of compound nouns which embody a very wide variety of grammatical forms, including relative clauses,

possessives, and noun-preposition-noun constructions*. He concludes that virtually any English grammatical relation can be embodied in a compound, and in addition, "there are unusually great opportunities for grammatical ambiguity in this kind of construction."

6.2.2. Elements of a new analysis

The bulk of Chapter IV of Lees' book is dedicated to the exposition of the grammatical types of compounds and their subclasses, and it is to this portion of his work that we look for possible guidance for a program attempting to understand the bibliography topics discussed above. Unfortunately, Lees' account is in terms of a transformational grammar, and is purely generative -- it has nothing to say about the recovery of underlying structure from the compounds**.

How, then, is all of the information to be recovered that is deleted in the many transformations that Lees has devised? Here I shall

* Lees has captured an impressive range of grammatical forms on pp. 118-119 of his book. Here are just a few examples:

puppydog	(= dog which is a puppy)
bulldog	(= dog which is like a bull)
shepherd's dog	(= a shepherd's dog)
watchdog	(= dog which watches something)
police dog	(= dog used by the police)
prairie dog	(= dog which inhabits the prairie)
hunting dog	(= dog with which one hunts)
blackbird	(= bird which is black)
howling monkey	(= monkey which howls)
night owl	(= owl which flies at night)
riding horse	(= horse for riding)
fishing village	(= village in which they fish)
laughing gas	(= gas which causes laughing)
baking powder	(= powder for baking (with))

** "Our analytic task is, then, to provide reasonable mechanisms in the grammar for the generation of a large variety of grammatically different nominal-compound types." [Lees 1963, p. 119, underline mine]

suggest, and investigate, the possibility of representing the underlying relationships of nominal compounds in a knowledge structure which explicitly accounts for the complex of information tying together the elements of the compound. If we can mirror Lees' analysis of grammatical types in our network representation, we will have a way to tie the compound elements into all of the other knowledge embodied in the net. SI-Net representation gives us, in fact, a broader range of possibilities than Lees had at his disposal -- dattrs can express any of many kinds of "intimate" associations between nominals. I will show how an analysis using dattrs brings out the underlying structure of these compounds, and yields, rather than a large number of seemingly arbitrary syntactic categories, a small number of conceptual structures. This analysis will illustrate how only two types of compounding operation can account for the entire spectrum of Lees' classes.

In addition, the uniformity of SI-Nets for representing verbal concepts as well as nominal ones will allow us to include in the analysis nouns created from verbs*. This makes the new classification consistent across all nominals -- not just "pure" nouns. It is to this type of nominal derivation that I now turn, before showing how to express compounding in terms more amenable to storing, associating, and retrieving bibliography references**.

* Since the initial writing of this thesis, it has been pointed out to me by Brian Smith that links for nominalization of verbal concepts may not be of the same, epistemological, type as, say, "DATTRS". I have still not resolved this to my own satisfaction; each of the nominalization links can be interpreted as a different type of structured inheritance. This may, in fact, be the underlying relationship between nominal and verbal concepts, and is something about which current ideas in linguistics are in flux: see the Appendix (Section A.2) for a hint of the "transformationalist"- "lexicalist" debate.

** This account of nominalization is fairly extensive. If interested mainly in compounds, skim Section 4.1.3, and turn directly to Section 6.4.

6.3. Deriving nouns from verbs

An understanding of the broad range of nominal expressions and compound-types that occur in document topic descriptions first requires an appreciation of the verbal sources of many of our English nouns. As I have discussed, in order to understand a compound, we need to determine the appropriate underlying relationship that exists between its two elements. Such relationships are very often based on verbal forms -- as we saw, Lees first postulated compounds to be derivable from NVN sentences, and as we shall see in Section 6.4, his subclasses are all based on sentence constructions centered around relations that appear in the verb. And, in fact, many of these verbal relationships are explicitly present in elements of the compound. For example, while we must infer the STEAL relationship in "car thief", the verb itself is present in a compound like "car owner". But notice that the verb does not appear in "pure" form. This, and a multitude of other compounds display their underlying verbal relationships in nominalized form.

Lees' compound breakdown depends intimately upon a detailed study of these "noun-like versions of sentences" [p. 54] that he presents in The Grammar of English Nominalizations. I will here deal briefly with a few of his more important nominalizations. However, it should be noted that his explanation of the nominalized forms that appear in compounds like "car owner" is purely syntactic. In addition, the subtypes of nominal expression that I will be dealing with are not so clearly circumscribed as Lees would have us believe -- Fraser [1970] and Chomsky [1970] both express alternative views of certain types of nominals. In an Appendix I attempt to sort out these somewhat confusing (when taken together) transformational accounts; but the goal of the eventual understanding of compounds by computer demands a different level of explanation. I will thus proceed immediately from a brief summary of the syntactic account of nouns derived from verbs to an attempt to understand the conceptual underpinnings of nominalization. To this end, I will reclassify some of

the nominal types in our conceptual notation, and show how to express the important ties between derived nominal concepts and their source verbal nodes. The uniform SI-Net representation of verbal relationships and nominal concepts allows the derived nominals to inherit attributes from the verbs, thus paving the way for a powerful, uniform method for representing compound concepts.

6.3.1. Agents, facts, and actions

Of the many kinds of nominals that Lees discusses, I will here concern myself with only four: the Agentive, Factive, Action, and Gerundive nominals. These should be sufficient to illustrate the power of the conceptual structure. In addition, I will introduce the Substantive, or Result nominal discussed by Fraser [1970].

The simplest of these is the Agentive -- a name for the agent of an action. The Agentive is generally created with the "-er" morpheme, producing familiar forms like "drummer", "owner", "lover", etc.

Another nominal discussed by Lees is the Factive. A Factive nominal is a reference to the fact that an event happened. With it we can make statements about the fact of the event rather than the way it proceeded. For example, "that he left was obvious" talks about the event as a whole rather than how its activity took place. The Action nominal, on the other hand, refers to the activity itself -- "his drawing was always done left-handed" is a statement about the action that took place during the activity. Action nominals, according to Lees, come in two forms: the "-Ing" form and the "-Nml" form. The -Nml nominals are basically all of those that do not end in "-ing"; these nominals can be abstract (e.g., "repair", "conservation", "control" -- these represent the general activity, and cannot take a singular determiner) or concrete (e.g., "test", "report", "attempt" -- these represent single events or objects, and often have singular determiners). In combination with

their direct objects, the Action nominals take an "of" -- for example, note the intervening preposition in "the retiring of his number" and "the retirement of his jersey".

Lees also presents the Gerundive, a nominal that always ends in "-ing", but which does not take any intervening "of". In "his driving the car surprised me," "driving" is a Gerundive, and refers to the fact that he drove. Contrast this with "his driving of the car gave me motion sickness," in which the same word is used as an Action nominal.

Finally, the Substantive nominal represents a separate entity produced as a result of some action. The use of the term "drawing" in "he owned a 5' by 3' drawing of Bobby Clarke" illustrates how the Substantive is different from both the Action and Gerundive nominals.

See the Appendix for further details on these nominal types.

6.3.2. Nominals conceptualized

The question we wish to ask now is, how can we express the conceptualizations underlying the syntactic analysis of English nominalizations? First, from what conceptual foundation do we start? We begin with concepts represented by nodes, as developed in Chapters 4 and 5; in particular, we are interested in structured concepts representing verbs.

The underlying representation for a verbal concept is that of the relation, the intension of a predicator of degree greater than one*. A verb would thus be represented as a node with a number of associated dattrs, which would represent its "cases". For example, Fig. 6.1 illustrates the verb "to hit" and some of its dattrs.

* Note that a property also bears a relation to the verb "to be" -- RED(X) can be written as "X is red." I will return to this shortly (the analysis here is independent of the degree of the predicator).

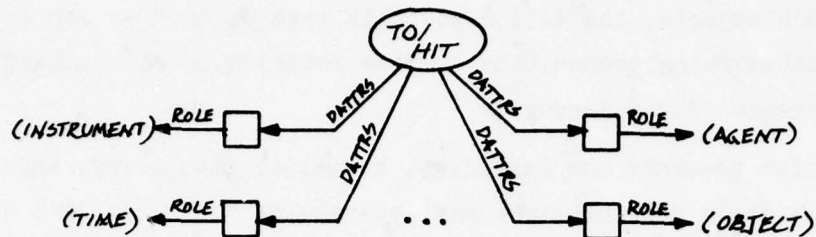


Figure 6.1. A simple verb structure.

The other form of the verbal concept that proves useful here is the event, an individuated version of a concept like TO/HIT. The node for "Carl hit Austin with my hockey stick this morning" would be a particular case of the general action, TO/HIT, and would have the associated role slots filled in accordingly. Recall that the connection of a particular individuator to its generic parent concept is accomplished with an INDIVIDUATES link and a mapping of the dattr, as in Fig. 6.2.

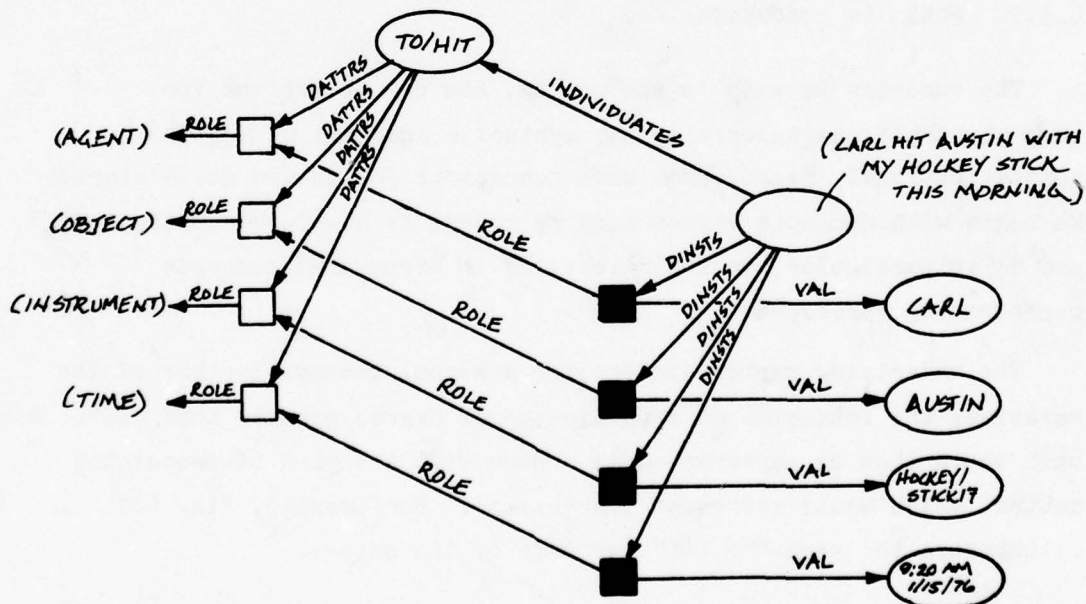


Figure 6.2. An event.

On top of this simple conceptual foundation, we can lay the main distinction to be drawn from the work of Lees, Fraser, and Chomsky -- nominals representing facts vs. nominals representing activities. It should be evident immediately that "facts" only pertain to particular events or actions. While there are several possible surface manifestations of the Factive nominal, we can capture the underlying generalization by having each of these references to the fact that the event occurred point in the same manner to the node for the particular event. Figure 6.3 illustrates a convention that we might follow to map

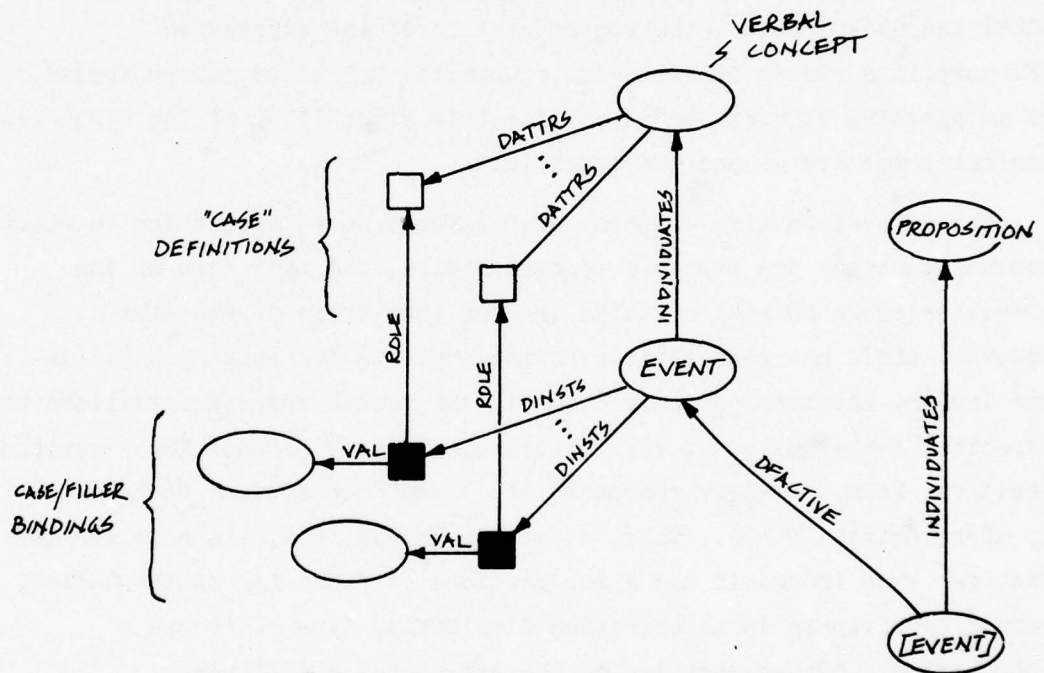


Figure 6.3. A Factive derived from an event.

out our nominalizations schematically: a link representing the particular type of nominalization will connect the source verbal concept to a node that represents the nominalized version of that verbal entity (the node labelled "[EVENT]" in the figure*). What we are doing in this

* This notation comes from Quine: "... we might adopt simply the brackets without prefix to express abstraction of medadic (0-adic) intensions, or propositions; thus '[Socrates is mortal]' would amount to

section, then, is determining the inheritance properties of such links and the characteristics of the derived nodes.

Returning to the DFACTIVE case in hand, we see that since the EVENT node is fully individuated, there are no open dattr's for the [EVENT] node to inherit, restrict, or particularize. The derived node should inherit all of the instantiated roles, however, so that the different syntactic forms manifested by the node will have the proper information with which to work. For instance, if a node for [Fonzarelli rode his motorcycle] did not inherit, by virtue of the DFACTIVE link*, the filled AGENT and OBJECT roles, the representation of the expression "Fonzarelli's riding his motorcycle appalled us" could not be formed. (I am agreeing with Fraser in calling this "fact" form of the Gerundive nominal a Factive -- see the Appendix.)

The derived Factive concepts can be found in relationships in which nominal concepts are expected to participate, and they take on the characteristics of such entities (rather than those of verbals). However, there are restrictions on the contexts for these nodes. In particular, Factives can only occur in the places where propositions are expected, for example, as the objects of verbs of propositional attitude (believe, know, promise, remember, etc.) and "non-action" verbals (prefer, detest, etc.). Thus, as we see in Fig. 6.4, the node for the Factive, even though it has a surface form of "what lay on the table", cannot participate in an operation like OWNING (you can't own a proposition). "I own what lay on the table" has a different

the words 'that Socrates is mortal', or 'Socrates's being mortal' when these are taken as referring to a proposition. It will be noted that in conformity with modern philosophical practice I am using the term 'proposition' not for a sentence, but for an abstract object which is thought of as designated by a 'that'-clause." [1960, p. 165]

* This is the reason that the link points from the nominalization node to the EVENT concept. This indicates the source of the inheritance in a manner similar to DSUPERC, INDIVIDUATES, etc.

Section 6.3.2
Nominals conceptualized

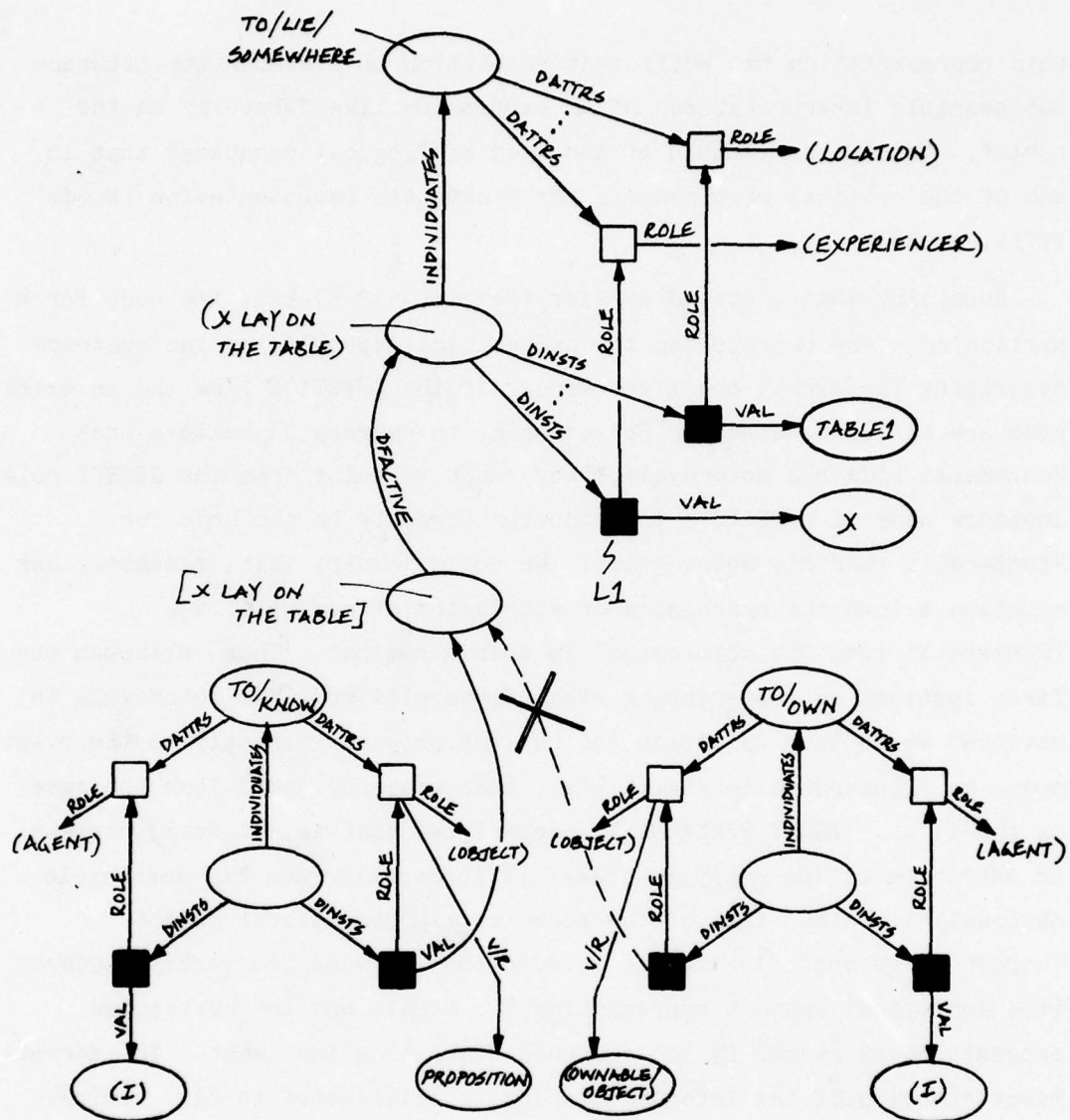


Figure 6.4. Contexts for Factives.

(non-Factive) underlying representation, that expresses that I own a particular object which happens to be lying on the table*. Notice how

* In this case, the OBJECT role of an OWN assertion would be filled by the particular entity that is owned (X in Fig. 6.4). To represent the detailed structure of the underlying relative clause (i.e., "the thing that lay on the table"), we would have the OBJECT role of the OWN

this representation has sufficient resolution to disambiguate between two possible interpretations of an expression like "what lay on the table". This is an example of the kind of "logical adequacy" that is one of the critical requirements for a semantic representation [Woods 1975a, p. 45].

Recalling that I stated earlier (Section 5.3.5) that the node for a particular event represented the proposition expressed by the sentence describing the event, one might wonder if the DFACTIVE link and an extra node are at all necessary. For example, to express "I believe that Fonzarelli rode his motorcycle," why can't we point from the OBJECT role instance node of a BELIEVE individuator directly to the node for "Fonzarelli rode his motorcycle"? We could, except that, remember, our notation allows the expression of attributes of nominals, and [Fonzarelli rode his motorcycle] is such a nominal. Thus, although our first instinct in representing "That Fonzarelli rode his motorcycle is obvious" would have us attach the OBVIOUS property directly to the event node, as illustrated in Fig. 6.5(a), that property would look the same as the < . . . AGENT FONZARELLI> property -- that is, it would express an attribute of the activity itself (*"Fonzarelli rode his motorcycle obviously"), rather than of the event as a propositional object. Therefore, we must distinguish between the individuated verbal concept (the individual concept representing the event) and the abstracted proposition expressed by the sentence describing the event. The correct representation of the intended meaning is illustrated in Fig. 6.5(b).

One other version of the conceptual Factive must be accounted for. Figure 6.6 briefly illustrates the concept of THUMB1, which is a thumb that is green. How can we derive from this representation a representation for "That THUMB1 is green is true"? Well, it should be

assertion point to node L1 in the figure, indicating the way that X is being described by virtue of its position in the TO/LAY/SOMEWHERE assertion.

Section 6.3.2
Nominals conceptualized

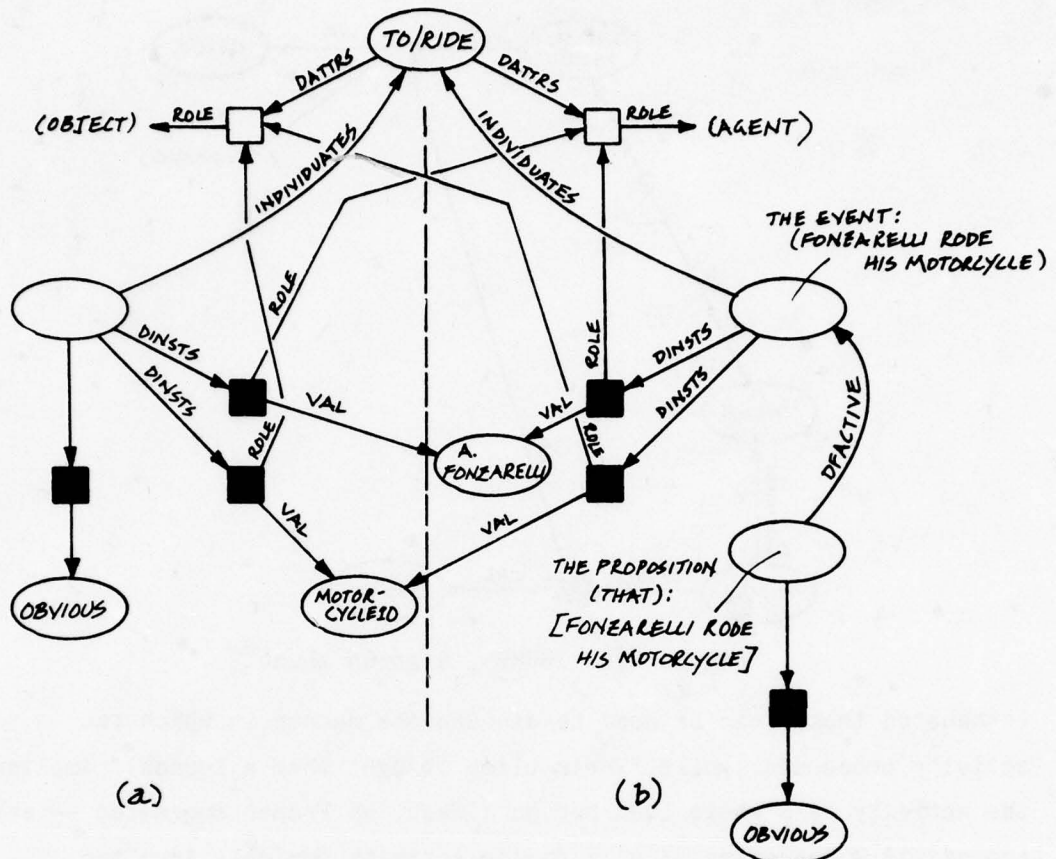


Figure 6.5. Properties of Factives.

clear from the intent of dattr's that they, too, have derivable Factives, since a role instance node represents the proposition that the <ROLE> of <CONCEPT> is <FILLER> (e.g., the AGENT of FONZARELLI/RODE/HIS/MOTORCYCLE is FONZARELLI). Thus, we can create the same kind of propositional abstraction that we saw above from role instance nodes. Notice, as Fig. 6.7 illustrates, this creates an interesting embedded representation.

The other nominals mentioned in Section 6.3.1 (except for the Substantive and the Agentive) deal with activities themselves. There seem to be two main kinds of what we might term the "Activity" nominal, one which deals with the ongoing process of the activity and one which designates the completed action. For example, "Their climbing

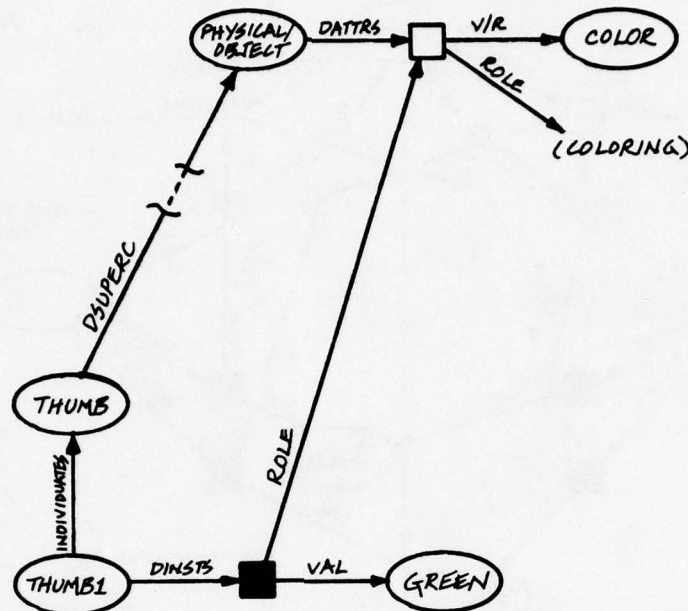


Figure 6.6. THUMB1, a green thumb.

(exhausted them)" can be used to discuss the manner in which the activity proceeded, while "Their climb (taught them a lesson)" implies the activity as a whole (but not as a fact, as Fraser suggested -- see Appendix)*. Therefore, I will divide Activity nominals into two classes, represented by the links DACTIVITY/PROCESS and DACTIVITY/COMPL-ACTION. In general, the syntactic manifestations of these nominals include the "of" before direct objects. In addition, PROCESS-type nominals usually end in "-ing", while COMPL-ACTION forms almost never do. But as Fraser ([1970] -- see the Appendix) has pointed out, exceptions exist in both cases.

Activity nominals can be derived from both general verbal concepts and particular events. In the latter case, such a nominal represents an

* Another type of nominal refers to the product of some activity; thus we have, in addition to "their cooking proceeded slowly," "their cooking tasted awful." The latter use of "cooking" is the Substantive; I return to this at the end of Section 6.3.2.

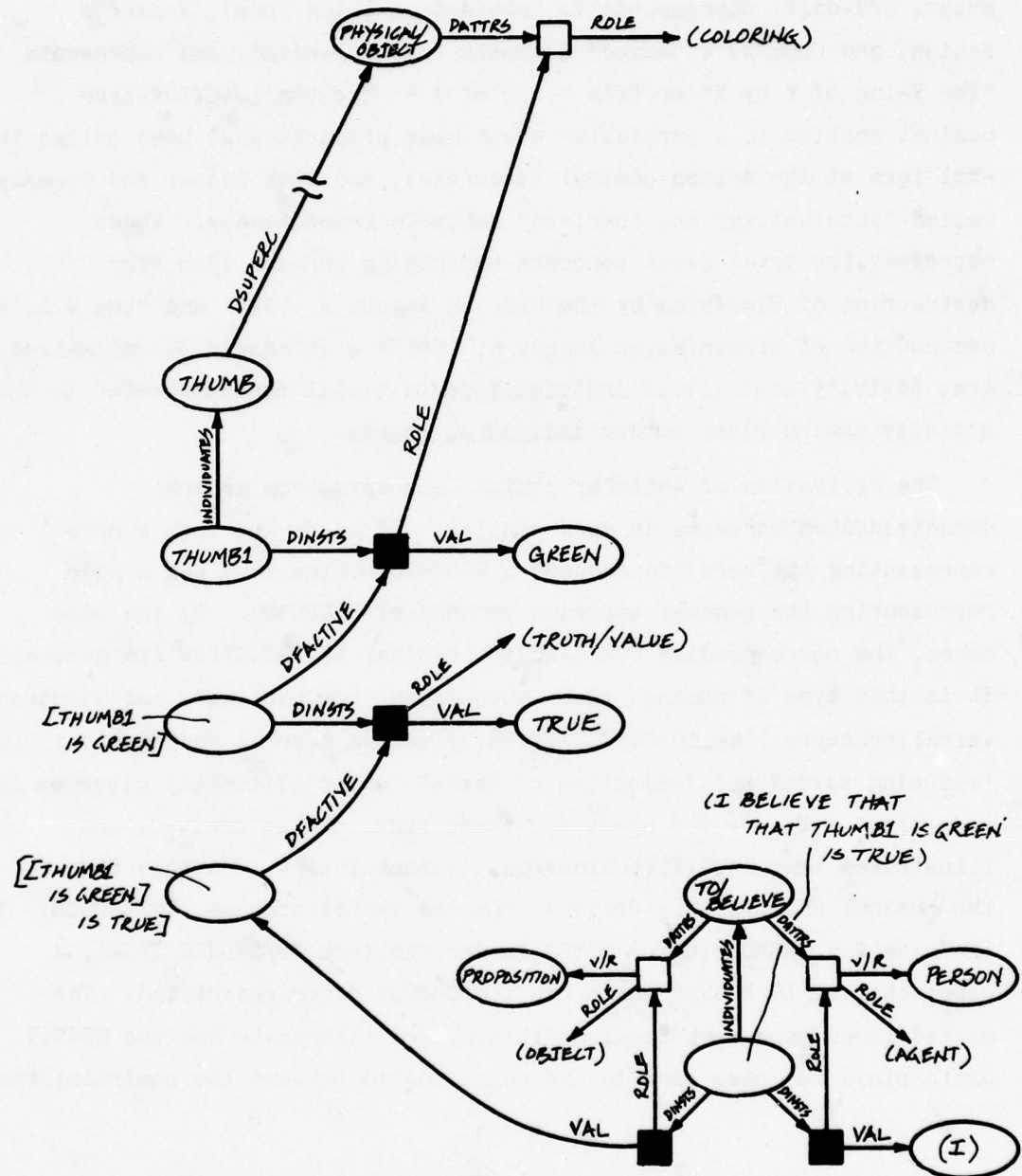


Figure 6.7. A Factive from a role instance.

abstraction of the activity that took place during the given event as a process over time or as a fait accompli. The PROCESS-type nominal of an

event, X/V-ed/Y, corresponds to Lees Action (-Ing form), Fraser's Action, and Chomsky's "mixed" nominals (see Appendix), and represents "The V-ing of Y by X" or "X's V-ing of Y." The COMPL-ACTION-type nominal applied to a particular event node produces what Lees called the -Nml form of the Action nominal (concrete), and what Fraser and Chomsky called "substantive" and "derived" nominals respectively. These represent the total event concepts underlying phrases like "the destruction of Hiroshima by the U.S. on August 6, 1945" and "the U.S.'s destruction of Hiroshima on August 6, 1945"*. It should be emphasized that Activity nominals of individuators of verbal concepts refer to the activity taking place during individual events.

The derivation of Activity nominal concepts from general unindividuated concepts is more complex. If we derive from a node representing the verb "to reduce" a PROCESS nominal, we get a node representing the general abstract process of REDUCING. By the same token, the corresponding COMPL-ACTION nominal is REDUCTION (in general). It is this type of nominal that, when taken from partially individuated verbal concepts like TO/REDUCE/TAXES, gives us general abstractions like "reducing taxes" and "reduction of taxes" (which ultimately gives us tax reduction). Figure 6.8 shows how these nodes can be derived, and illustrates that DACTIVITY links pass dattrs intact. In this figure, the nominal REDUCTION is derived from the verbal concept, TO/REDUCE. In the same way, REDUCTION/OF/TAXES is derived from TO/REDUCE/TAXES, a subconcept of TO/REDUCE which has its OBJECT dattr restricted. The dotted lines represent the inheritance, and illustrate how the OBJECT dattr plays the same part in the relationship between the nominals; the

* It is possible in some idiolects to say, "The destruction proceeded painfully slowly," which refers to the activity rather than the event as a whole. Thus the underlying structure is not strictly determined by the surface form of the nominal, but by context as well. I am here concerned with the underlying structure, and am using the most common surface forms only for illustrative purposes.

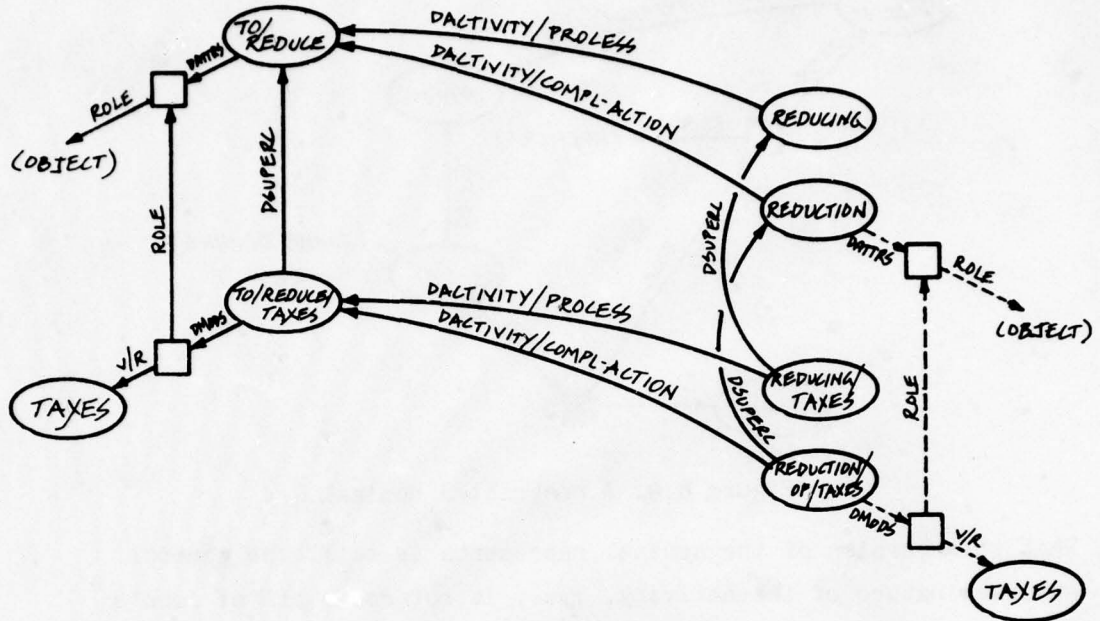


Figure 6.8. Derived abstract nominals.

dattrrs of the defining verbal concept are available at the nominal nodes for modification and individuation in the normal way. This provides a general facility for producing meaningful restricted versions of nominals.

So far, the Activity nominal derived from a generic verbal concept looks like that derived from a particular event, except for the fact that it describes a general abstract activity rather than a single event's activity. There is an important variation on this Abstract version of the Activity nominal that can be illustrated by contrasting some common surface forms -- let us consider how the interpretation of the surface manifestation of the abstract Activity nominal changes with the addition of a determiner. First, we might have a subject with a possessive morpheme, yielding, say, "John's driving". This addition does not really alter the type of nominalization, since what it does is fix only the AGENT role of the general concept, just as the OBJECT role filler was fixed in the above case, "reducing taxes" (see Fig. 6.9).

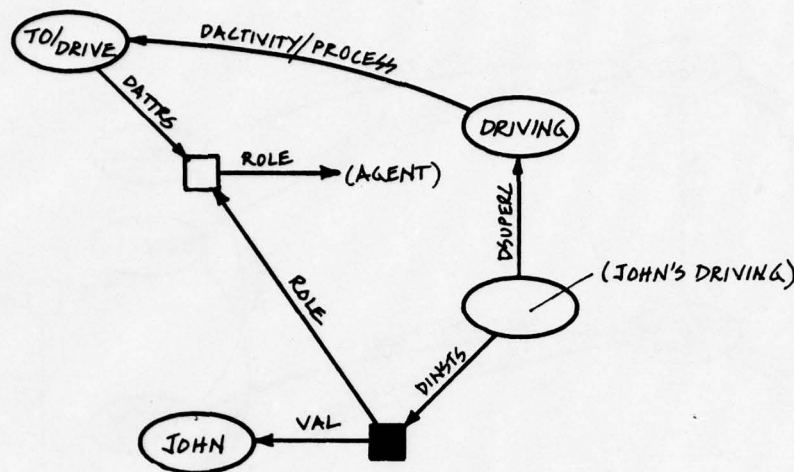


Figure 6.9. A restricted nominal.

What this version of the nominal represents is still the general abstract nature of the activity, i.e., it refers to all of John's driving collectively. This nominal allows us to make a statement like, "John's driving is atrocious," without necessarily implicitly impugning any particular instance of his driving*. This is the essence of the Abstract version of the Activity nominal -- it refers to a group of potential events as a general kind of activity, not as a set of discrete events. An instance of it still defines a general kind of activity, not the activity of a particular event. For example, "John's climbing of trees on Tuesday" refers to all of his climbing activity on that day, not a single climbing of a single tree (which would be a single event).

On the other hand, if we add a singular determiner like "a", as in "a meeting of minds" (or "a single climbing"), we produce a truly different nominal than the Abstract. This one stands for a single

* Fraser, as mentioned in the Appendix, claims that this is a Substantive nominal. I disagree -- consider "drawing" instead of "driving". I might say that "John's drawing is meticulous," meaning the way he does it, or that "John's drawing is a life-size portrait of Darryl Dawkins," obviously referring to a different "drawing". The first is similar to the case above, the second is a Substantive.

event, not an abstract general activity. This is just like the DACTIVITY nominal derived from a particular event, except for the fact that we do not know which particular instance it is. An individual event of this type could be described as one of these ("Theirs was a true meeting of minds"), but taken alone, this nominal is indefinite and singular. We will call this a Generic form of the Activity nominal, as it defines the structure of a single event rather than a composite of all activity of the same nature (which the Abstract form defines). It will be represented by a "DGEN" link to the Abstract nominal node from the node for the Generic form. We will return to this in a moment.

The definite determiner, "the", has a more context-dependent effect -- it can produce either the Abstract or the Generic form. If the object of the nominal is indefinite, then the resulting nominal is still abstract, just as was "John's driving". For example, "the driving of cars" is an abstract reference to that kind of activity in general. Notice that "The driving of cars is prohibited" deals with the notion as a whole, and is virtually the same as "Driving cars is prohibited." In addition, this same kind of statement can be made in another way, also using the Abstract nominalization, in this case, "No driving of cars is permitted."

A close look at the use of the nominal, "driving", in this last case will help to highlight the difference between the Abstract and Generic forms. In "No driving of cars is permitted" we refer to the general activity of driving (i.e the Abstract form). Contrast this with, "No climbing of Mt. Everest has been attempted." In this case, we are saying that no single instance of the general class of climbings has existed. The definite, "Mt. Everest", unlike the plural, "cars", permits the interpretation of the phrase as what I have called a "generic" concrete event. The Generic refers to a kind of event, so that "No climbing of Mt. Everest . . ." means that no climbing events (i.e., "climbings") have occurred. The Abstract refers to a kind of activity, so that "No driving of cars is permitted" means that no

driving activity is allowed.

The Generic version of the Activity nominal is a different type of entity than the Abstract version. Its appearance is singular, while the Abstract version refers to a collection, or "mass" of activity. The Generic is a way of talking of all events of a given nature by referring to a single, paradigmatic, abstracted version of the event. For example, "The swimming of the English Channel is an arduous undertaking" entails that every swimming of the Channel is arduous. This, recall, is very similar to the meaning of a standard concept in SI-Net formalisms -- an abstract entity that implicitly stands for a class (i.e., the class of all extensional entities described by the concept), yet has the form of a single generic member of the class. The Abstract form refers to all events of a given nature by describing them en masse.

With this dichotomy in mind, we see that the Activity nominal applied to a particular event gives us an individuator of the Generic form of the nominal, rather than the Abstract form. As such, the nominalized form of an event like "The U.S. destroyed Hiroshima on August 6, 1945" would be derived not by a single DACTIVITY link, but by that link followed by a DGEN link. The node between the event and the Generic nominal (i.e., the one pointed to by the DACTIVITY link) seems to have no English counterpart.

The surface-form examples of the Activity nominal given above cannot be taken too seriously, since it is easy to find ambiguous or non-conforming examples. For instance, "A swimming of the Channel takes courage" is Generic and describes how all swimmings take courage, while "A meeting of minds was held" is an indefinite instance of the Generic, and isolates only a single event. On the other hand, "No swimming" indicates the general abstract swimming activity, while "A meeting of minds is tiring" generically describes all such meetings. We will consider these pairs of nominals to refer to different underlying conceptual structures, even though their surface forms are identical. In any case, it should be clear from the discussion that there are three

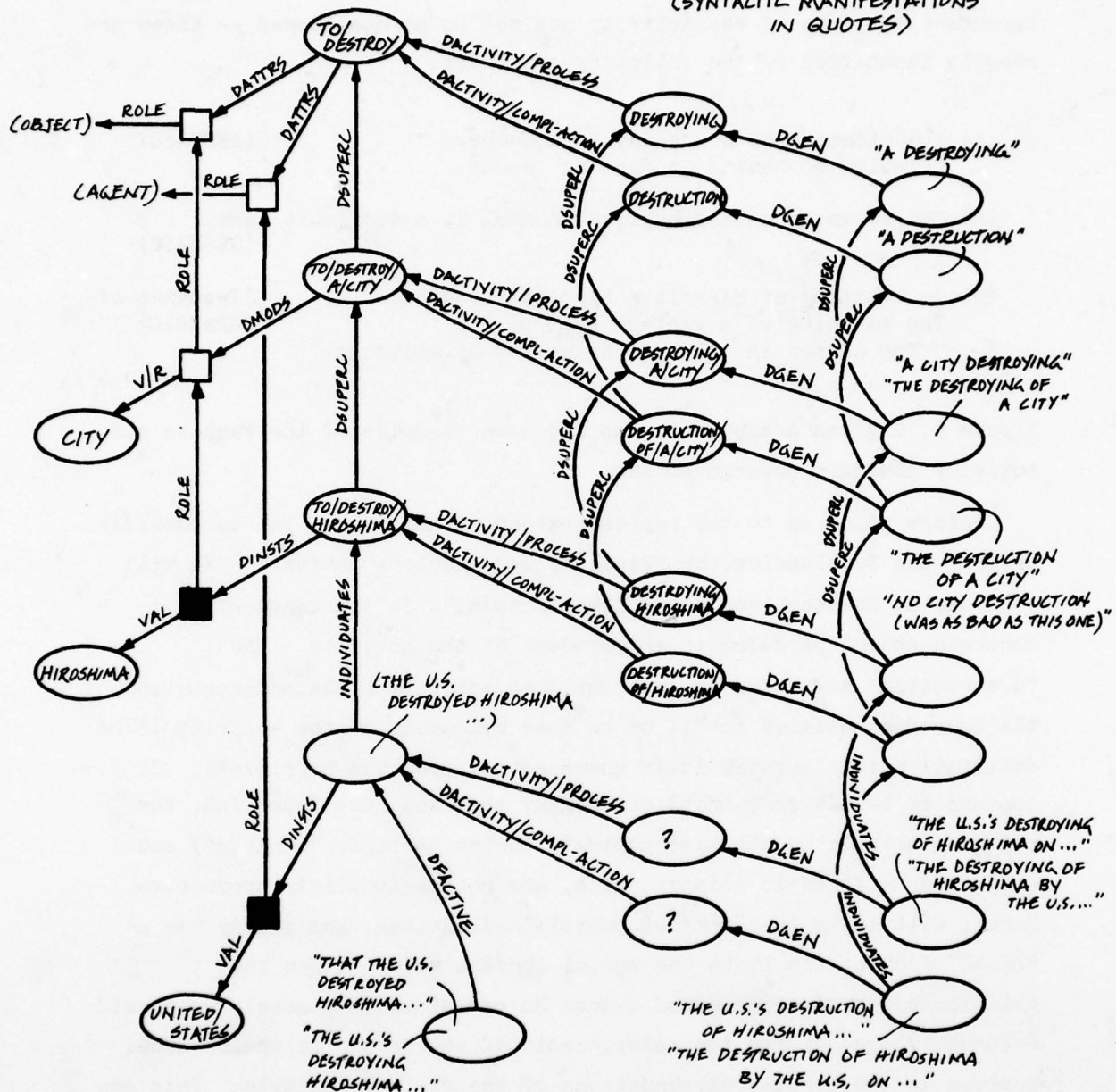
Section 6.3.2
Nominals conceptualized

important subtypes of the Activity nominal to be considered -- these are roughly identified by the following sentences:

- (a) The climbing of mountains is arduous. {ABSTRACT}
Climbing mountains is for the insane.
- (b) The swimming of the English Channel is a difficult task. {GENERIC}
- (c) An orbiting of Mars is scheduled for 1981. {Instance of
The pitching of a perfect game by GENERIC}
Don Larsen in 1956 was witnessed by millions.

Figure 6.10 gives a schematic map and some examples of the Factive and Activity nominals covered so far.

Before going on to the representation of compounds, let us consider briefly the Substantive (or "Result") and Agentive nominals. We will consider as Substantives nouns like "drawing", in the sense of a concrete object produced as a byproduct of the activity. Thus, "destruction" may refer to the completed activity ("Their destruction of the town was uncalled for"), or to some byproduct of the activity ("The destruction that greeted their unsuspecting eyes was horrible"). It appears to be the case that, as Chomsky proposes (see Appendix), the relation between Substantive nominals (which he calls "derived") and their source forms is idiosyncratic, and not particularly productive. Here I will leave this nominal unexplained further, and simply use a DRESULT link to tie it to the verbal concept node. Since the relationship thus represented cannot be explained in general, we should eventually account for the relationship of such nouns to their verbal sources in the structural conditions of the derived concepts. This can be done in a manner similar to those described for HYDROGEN/BOMB and MESSAGE in Chapter 5, in which cases nouns were defined in terms of the operations on them. I will not pursue Substantives here except to mention that they, too, seem to inherit dattr's freely from their defining verbals.



Agentives exhibit more regular behavior in their relationships with defining verbals. They can be thought of as being derived not from the verbal concept node, but from its AGENT role description node. The

Agentive inherits dattrrs from the source in a uniform manner, although not all are conventionally useful. For example, the OBJECT role of an Agentive derived from a transitive is almost always usable to create subclassifications of agents like CAR/OWNER, SHOE/MAKER, VW/MECHANIC, etc., but only rarely are locatives or manner dattrrs used (GARAGE/MECHANIC, CAT/BURGLAR). I will illustrate many examples of this kind of nominal and its inheritance characteristics in the next section.

Figure 6.11 sketches how a DROLE link might be used to express the

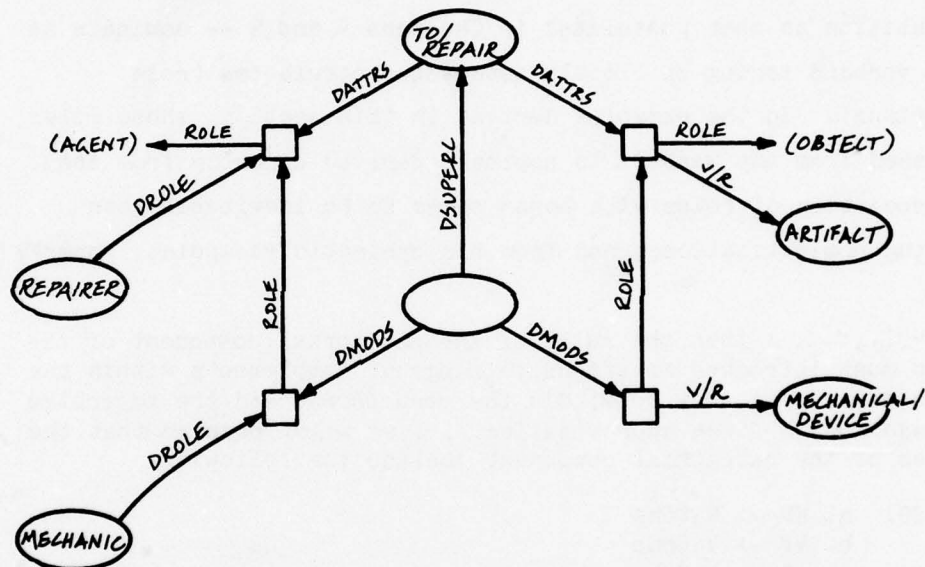


Figure 6.11. Derived role nominals.

derivation of the Agentive nominal from the AGENT role. As just noted, the source of the dattrrs to be associated with the Agentive is to be located by following the DROLE link, and then the inverse of the DATTRS link. All dattrrs of the concept thus found, except for the one initially traversed, are accessible, producing restricted Agentives like "repairer of shoes" (or, ultimately, "shoe repairer") and "VW mechanic".

I should briefly mention that the derivation of this type of nominal suggests a possible generalization. It may not be unreasonable to attempt to derive other nominals from other dattrrs of verbal concepts.

For example, the verbal context TO/ENTRUST gives rise to the well-known concept, TRUSTEE, which exhibits a similar type of behavior in relation to ENTRUST as PAINTER does to PAINT, except that it names the OBJECT role. By the same token, a "graduate" is one who is graduated, and the concept GRADUATE should be expected to inherit all dattr's but the OBJECT from the verbal concept (e.g., we have "1971 graduate", "graduate of Princeton", etc.). It is not clear whether other similar types of nominalized roles are used in English.

Notice that this treatment of nominals has produced the same basic representation as that postulated in Chapters 4 and 5 -- nominals as well as verbals taking on closely associated attributes (role descriptions). In the examples derived in this section, those roles were passed from the verbals to nominals derived directly from them. This association of roles with nouns seems to be inevitable when discussing nominalizations, and from his syntactic viewpoint, Chomsky states,

Clearly, . . . then the rules of the categorial component of the base must introduce an extensive range of complements within the noun phrase, as they do within the verb phrase and the adjective phrase. As a first approximation. . . we might propose that the rules of the categorial component include the following:

- (20) a. NP -> N Comp
- b. VP -> V Comp
- c. AP -> A Comp
- (21) Comp -> NP, S, NP S, NP Prep-P, Prep-P Prep-P, etc.

Is there any independent support, apart from the phenomena of derived nominalization, for such rules? An investigation of noun phrases shows that there is a good deal of support for a system such as this.

Consider such phrases as the following:

- (22) a. the weather in England
- b. the weather in 1965
- c. the story of Bill's exploits
- d. the bottom of the barrel
- e. the back of the room
- f. the message from Bill to Tom about the meeting
- g. a war of aggression against France

- h. atrocities against civilians
- i. the author of the book
- j. John's attitude of defiance towards Bill
- k. his advantage over his rivals
- .
- .
- .
- w. a nation of shopkeepers

In each of these, and many similar forms, it seems to me to make very good sense -- in some cases, to be quite necessary -- to regard the italicized form as the noun of a determiner-noun-complement construction which constitutes a simple base phrase. . . . [Chomsky 1970, pp. 195-196]

The structures (22), and others like them, raise many problems; they do, however, suggest quite strongly that there are base noun phrases of the form determiner-noun-complement, quite apart from nominalizations. In fact, the range of noun complements seems almost as great as the range of verb complements, and the two sets are remarkably similar. [Chomsky 1970, p. 198]

Thus the linguists have made the same kind of observation that I have embodied in dattrs. The notion has been made precise by virtue of its being embedding it in the SI-Net framework -- to some extent this represents a synthesis of the ideas of Fillmore [1968] on cases for verbs and the above suggestions by Chomsky*.

6.4. The structure of English compounds

As I have mentioned, nominal compounding is an extremely productive linguistic activity, and one particularly well-suited to the information-compaction task inherent in annotating bibliographies. Complex relationships between concepts introduced and discussed in a document may be abbreviated by appropriately stringing together sequences of nouns or nominalized verbs, thereby producing brief, but

* This attribution of roles to nominals has deeper implications with respect to Chomsky's recent "x-bar" theory, of which the above quote is a hint of a beginning. This looks like a fruitful research direction.

expressive topic descriptors for the document. While a tremendous effort-saver linguistically, compounding represents a very difficult problem for a computer program that tries to understand these topic descriptions. The relationship between compound elements must be inferred in order to properly interpret the complex of concepts underlying the linear string of nouns.

Given some existing structure representing what the system "knows" (call this the knowledge base), this type of inference process requires first the locating of the particular concepts which represent the nouns in the compound, and then the determining of the particular relationship most reasonable to expect between those concepts. We can, for purposes of the present discussion, assume the lookup process and focus on the determination of a reasonable conceptual relationship. This latter can involve either the picking out of some particular relationship that is already explicitly represented in the knowledge base, or the creating anew of a relationship not explicitly found there.

Both types of relationship determination depend fundamentally upon the structure used to represent the concepts. The syntax of the data structure for concepts constrains in advance the general forms that all potential representations can take*. Thus all of the relationships that might potentially be represented in the notation are strictly circumscribed by the rules for forming concept structures and the particular foundational knowledge expressed in these structures. That is, the syntax of the formalism predetermines the set of well-formed concept structures, and the initial set of concepts determines the semantically acceptable ones.

* For example, SI-Net notation determines in advance of all instantiations of it that no role description node will itself have dattr's, and that the only way that the notion of a role having dattr's can be represented is by a nominal node derived -- by a DFACTIVE or DROLE link -- from the role node.

Thus the success of the inference of a relationship between two nominals depends entirely upon the power of the formalism from which the knowledge base is constructed. If a notation captures only basic grammatical relations like subject, object, and prepositional ("oblique") object, then no finer or more subtle relationships than those can ever be inferred to hold between the elements of a compound. While statements such as these might seem tautological, it is often not appreciated that a particular representation scheme determines in advance the set of all "potential concepts" that it can handle. To reiterate the methodology statement of Chapter 3, it is this type of formal adequacy of a representation to which I wish to draw attention. I here want to produce a mechanism for representing "reasonable" relationships between concepts such as those found in nominal compounds. It must be kept in mind that our set of notation operations determines in advance all of the kinds of relations that we can ultimately represent.

In this section I look to the representation developed in this report as the structure for a knowledge base that might represent nominal compounds in a way more suitable to the bibliography task than Lees' transformational account. Recall that Lees' treatment was purely generative; in addition, his classes were structured according to relations like subject-object, verb-object, subject-prepositional object, etc. Neither of these features would help a computer program trying to understand the underlying conceptual structure of descriptions of bibliography references*.

* Lees admits to the shortcoming of his classification in a later paper [1970], in which he suggests that a case structure like Fillmore's would be more appropriate for analyzing the structure of nominal compounds than his own "antediluvian" account. While such a suggestion looks promising in its similarity to the one put forth here, it is not carried very far by Lees. In addition, the SI-Net "case" mechanism is more general than Fillmore's, and, as I have discussed (Section 5.1.3.1), the notion of a small number of fixed deep cases is difficult to support.

I will look at several of Lees' classes of compounds and present a more perspicuous underlying representation for them. I will thus examine in detail some particular examples of the general notation being developed here. However, rather than provide an exhaustive account of the structure of English compounds, I will attempt to illustrate how the representation at hand can handle some important representative cases. If this can be done convincingly, then we might infer that the structure has the right "handles" to adequately represent all of the kinds of conceptual relationships that underlie nominal compounds. The enumeration of all of the particular types of relationships is, of course, an open-ended task, left to the processing of many particular bibliographies with a detailed initial knowledge base. It is hoped that the particular examples presented here, coupled with the general framework underlying their structure, will prove adequate evidence of the power and appropriateness of this analysis.

6.4.1. Verb-plus-dattr compounds

Lees presents (among others) the following eight classes of compounds:

- (I) Subject-Predicate
- (II) Subject-"Middle Object"
- (III) Subject-Verb
- (IV) Subject-Object
- (V) Verb-Object
- (VI) Subject-Prepositional Object
- (VII) Verb-Prepositional Object
- (VIII) Object-Prepositional Object .

In three of these (III, V, and VI), the verbal relation appears explicitly in the compound (I will focus on classes III and V). This type of compound is a good place to begin the analysis, since the relationship is simply that of a verbal concept to one of its special roles, the AGENT or the OBJECT cases of the verb. I will first concentrate on Class V of Verb-Object compounds like "call girl" (a girl

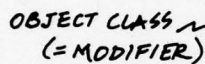
that one calls), "drinking water" (water for drinking), "bull fighting" (the fighting of bulls), "mail delivery" (the delivery of mail), "blood test" (a test of blood), and "Nixon hater" (one who hates Nixon).

Notice that these particular verb-object compounds are easily split into two groups: one with V-O order, the other with O-V order. According to Lees, the former gives us compounds with Infinitival and Gerundive nominals, the latter with Action nominals, which come in -Ing, -Nml (both abstract and concrete subforms), and -Er forms. The underlying structural representations of these forms, however, are extremely similar, with the main difference between O-V and V-O forms being which node of the pair is the superconcept of the node for the compound as a whole.

6.4.1.1. Object-Verb compounds

The basic conceptual structure of an O-V compound with the object preceding the verb is indicated in Fig. 6.12. The compound stands for some restricted nominalized form of the verb, whose OBJECT VALUE/RESTRICTION is more limited than in the general case (in general, these compounds are formed with restricted, but not particularized dattrs)*. For example, "news broadcasting" is based originally on the verbal concept, TO/BROADCAST. The DACTIVITY/PROCESS nominal gives us BROADCASTING (i.e., the abstract activity of broadcasting in general). This nominal has its OBJECT role restricted such that it may only be filled by those things which can be considered to be NEWS, thus producing a restricted Activity nominal which represents the abstract activity of news broadcasting. Figure 6.13 illustrates the precise structure of this compound. That the concept representing the compound

* In the figure and those to follow, the node representing the compound will be heavily inked, and each of the components will have an asterisk in its node.



* = ELEMENT OF COMPOUND


 = REPRESENTATION OF COMPOUND ITSELF

Figure 6.12. Basic O-V structure.



Figure 6.13. "News broadcasting".

itself is a type of PROCESS should be clear from the DSUPERC link to the nominalized version of TO/BROADCAST.

Lees separates compounds similarly constructed with the -Nml Action nominal into two subclasses -- "Abstracta" and "Concreta". In light of the conceptualized nominals that we introduced in Section 6.3.2, this difference is easily understood as the distinction between the Abstract form of the nominalized verb and the further derived Generic form. The latter represents a singular concrete event (of the type indicated by the verb) whose particular referent is indeterminate. Thus, we can easily capture compounds like "birth control" and "book review" within the paradigm of Fig. 6.12 (see Fig. 6.14). BIRTH/CONTROL is the general

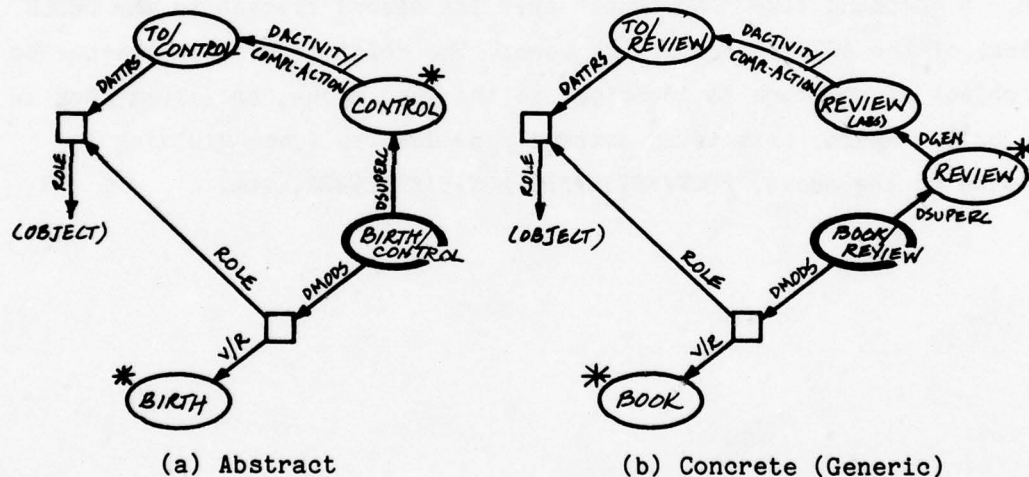


Figure 6.14. Abstract and concrete COMPL-ACTION compounds.

overall activity of the control of birth, while BOOK/REVIEW is a singular instance of the general activity of reviewing. We can have a book review, but not a birth control. The determiners and number of the object determine which sense one gets (or conversely, the sense intended determines what determiners one can use); in addition, some verbs, like

"control", do not seem to have Generic nominalizations*.

Notice the extreme productivity of this paradigm. We can have a single BOOK/REVIEW event, as well as the Abstract BOOK/REVIEWING. Either of these can be changed to MOVIE/..., THEATER/..., etc. In most cases, a meaningful compound can be made from any of the Activity (and Result) nominals of a verb paired with a restricted OBJECT dattr, and the restrictions can be as varied as there are nodes in the network which represent subconcepts of the original VALUE/RESTRICTION of the OBJECT**.

This productivity follows equally as well for the Agentive nominal, although its underlying form is slightly different from that of Fig. 6.12. A compound like "car owner" owes its second element to the DROLE nominal of the AGENT dattr of the verb. The relation of this nominal to the object of the verb is identical to the ones above, as illustrated in Fig. 6.15. Again, this is an extremely productive form, yielding in addition to the above, BOOK/REVIEWER, MOVIE/REVIEWER, etc.

* "Book review" has two possible senses -- here I mean the sense of the activity of the review ("Since no one in the class had read the text, we had a book review today"); one could just as well talk about the concrete object produced as a product of this activity ("He turned in his book review two days late"). This latter Substantive sense would be represented in a manner similar to that of Fig. 6.14(b), except that the DACTIVITY and DGEN links would be replaced by a single DRESULT link. This would indicate the derivation of the concrete "review" as a result of the activity.

** It is rare to find all forms in common use concurrently. However, I am trying to account for potential compounds and their interpretations, and this formal account does not try to anticipate in advance which concepts will attain popular use.

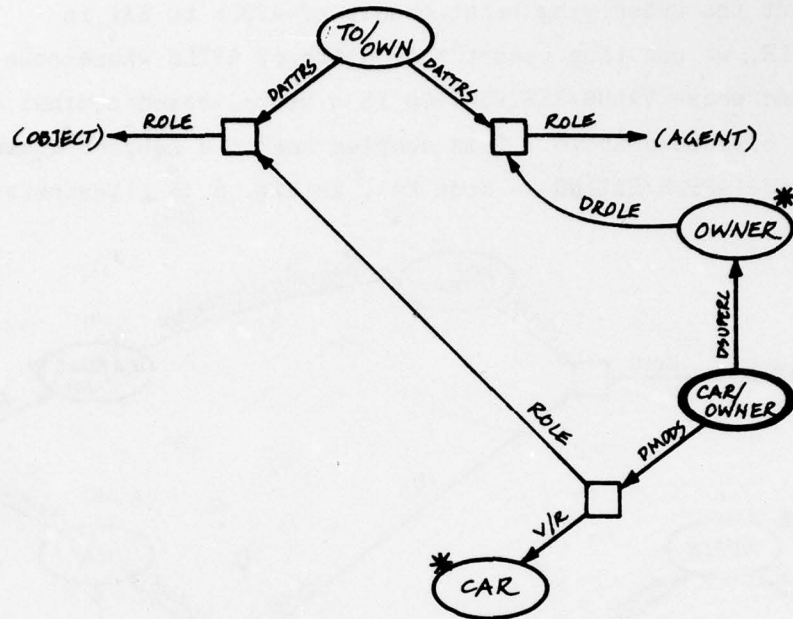


Figure 6.15. Object plus Agentive.

6.4.1.2. V-O compounds -- Habitual activity and purpose

The other compounds of the verb-object group, those with structures V-O, represent particular kinds of objects used in the activity named by the verb. These are presumably more restricted versions (i.e., subconcepts) of those concepts named by the second elements of the compounds. For instance, FARM/LAND is a particular kind of LAND, and EATING/APPLEs constitute a subclass of APPLEs in general. Lees' derivation of the structure of such compounds resorts to what he calls a "'purpose' adverbial", since "eating apple" is easily derived from the noun and a "for-phrase of 'purpose'": ". . . apple for eating ---> eating apple" [Lees 1963, p. 149]. This suggests a similar structure for the conceptual representation -- the difference between APPLE and EATING/APPLE is the habitual use attributed to the latter.

This notion of purpose, or habitual activity, seems a natural role to associate with nominals like those in "chewing gum", "drinking water", "riding horse", "punching bag", "draw string", "rip cord", etc.

AD-A056 524

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MASS
A STRUCTURAL PARADIGM FOR REPRESENTING KNOWLEDGE.(U)
MAY 78 R J BRACHMAN
BBN-3605

F/G 5/6

UNCLASSIFIED

N00014-77-C-0371
NL

3 OF 4

AD-A056 524
6624



6524

to represent the underlying relationship of APPLE to EAT in EATING/APPLE, we can thus resort to a dattr of APPLE whose role is PURPOSE, and whose VALUE/RESTRICTION is a verbal-based nominal exactly like those discussed above. This complex has as a subpart a complete -V compound (APPLE/EATING -- node EA), as Fig. 6.16 illustrates. The

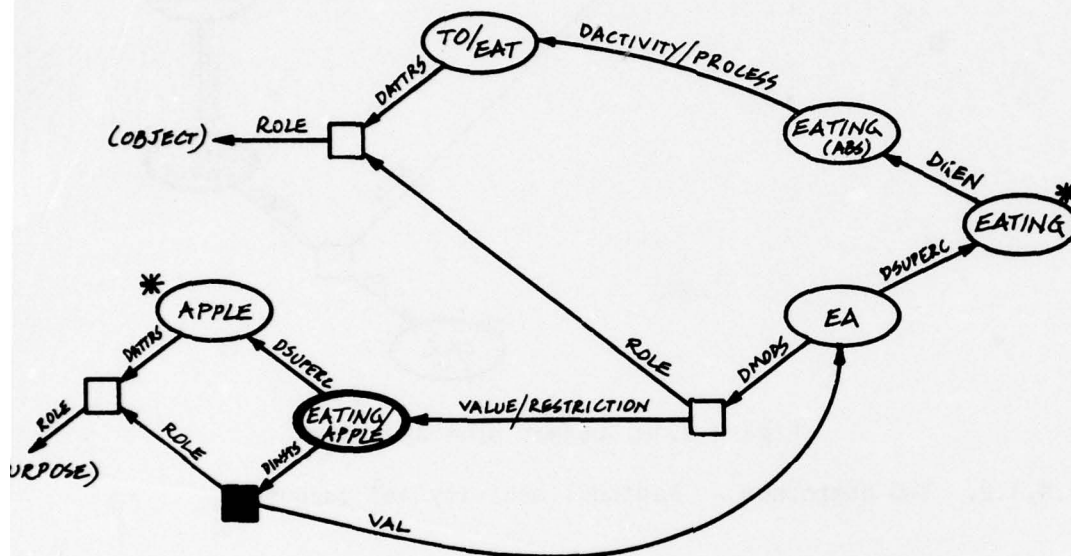


Figure 6.16. A "purpose adverbial" -- V-0.

structure mirrors in a more conceptual way Lees' derivation from "... apple which is for eating the apple", where the PURPOSE role expresses the "which is for ..." clause, and node EA expresses the "eating the apple" phrase. I have resorted to the Generic nominal here to indicate that the purpose of such an apple is the eating of that apple. As noted in Section 6.3.2, this singular form, but general nature, calls for a nominal different from the Abstract.

While infinitives in conjunction with activities have not been previously mentioned, we can account for Lees' "action" form of the infinitival nominal [1963, pp. 73-80] in a manner similar to our treatment of Gerundives. Lees has two forms of the Infinitival nominal, "action" and "fact", and we can reflect that division just as we did

with the -Ing form nominals. Thus we might postulate an Activity nominal with infinitival form (e.g., "To swim is invigorating") and a Factive nominal which can be derived from a general verbal concept (e.g., "For them to leave would be a pity"). The Infinitive type of Activity would then account for the class of compounds like "farm land", where the verb appears in uninflected form but stands for an activity to be done to the object which appears as head of the compound. The conceptual representation for the compound "farm land" looks virtually the same as it would for "farming land", being derived from "land to farm", similar to "land for farming".

An interesting feature to point out here is the consistency of our representation across syntactically differing, but semantically similar, compounds. It is clear that these compounds are dealing with activities, not facts (i.e., their purpose is for some activity to be carried out on the object), and while we might have several slightly different ways to express an activity (witness "farming land" as well as "farm land"), the underlying relation between the verb and object is always the same. This is mirrored in the SI-Net representation by the consistent use of a modified OBJECT dattr attached to slightly different nominal concepts (which hopefully reflect the subtle differences between the uses) associated with the verb in the compound. This consistency and apparent reflection of underlying structure can be contrasted with Lees' abandoning of the Action nominal and resorting to the Gerundive to explain these compounds [1963, pp. 149-50]. His explanation classes the modifier in "eating apple" as a Gerundive, yet the head of "apple eating" would be an Action nominal. This seems to be an unfortunate concession to the whims of syntax.

6.4.1.3. Subject-Verb compounds

This consistency carries over nicely to Lees' Class III, Subject-Verb compounds. These compounds are remarkably similar to those of Class V, the only significant difference being the obvious use of the AGENT dattr rather than the OBJECT. There seem to be no S-V compounds equivalent to the "news broadcasting" group (i.e., with a PROCESS form of the verb as second element), but there are S-V subtypes virtually identical to the COMPL-ACTION abstract and concrete classes above. For example, the representations of "food spoilage" (abstract) and "snake bite" (concrete) are just like those of "birth control" and "book review", depicted in Fig. 6.14 above (see Fig. 6.17).

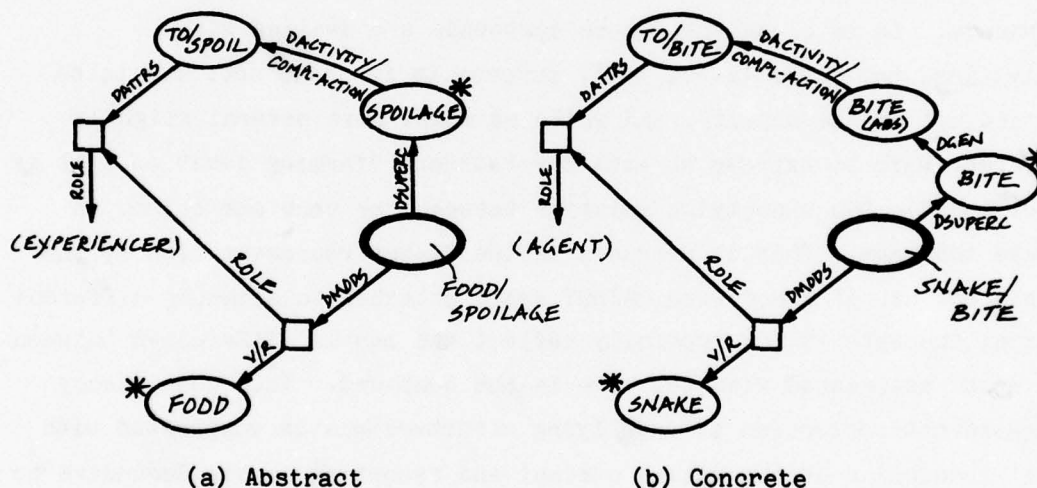


Figure 6.17. S-V compounds.

If we were rigidly to maintain this parallel, substituting only AGENTS for OBJECTS, we would naturally conclude that the Agentive nominal could not be used as head of one of these compounds, since the AGENT dattr is held down by the first element of the pair. But notice that in a compound like "food spoilage", the FOOD is not the agent of the SPOILING, but only the inanimate experiencer (unless, of course, the food is spoiling something else, like a party). As a result, a compound

like "food spoiler" is not impossible (perhaps "food spoilant" is more acceptable).

This important difference between EXPERIENCER and AGENT can be accounted for by simply differentiating between these roles in the underlying structure. Lees, however, cannot account for such differences, since his criterion is "subject", rather than a deeper case. He does to some extent distinguish between groups, using "of-periphrasis" and "by-periphrasis" to separate the subclasses. But these are not sufficient to capture the real dichotomy -- "onion smell" is grouped with "snake bite" (the onions obviously do not do the smelling), while "insect flight" and "dealer maintenance" are classed separately.

The reverse compounds, in which the verb is the modifier rather than the head, give us the same types of classes as did the V-O compounds, namely those like "talking machine" (cf. "eating apple") and "dive bomber" (cf. "farm land"). In addition, a class of COMPL-ACTION-type verbs can combine with their agents to form pairings like "assembly plant" and "suction pump". The former two classes ("talking machine" and "dive bomber") are structured like their verb-object counterparts, with the sense of habitual action being added to tie the agent to its corresponding compound; the latter is similarly structured, as seen in Fig. 6.18*. The sense of habitual action is important, since, for example, a "wading bird" (where underline indicates stress) is a bird who happens to be wading right now, while a "wading bird" is one whose nature it is to wade, and is not necessarily anywhere near water at the

* No Generic is needed here, since the OBJECT of the ASSEMBLY is still indefinite, and thus the sense of "assembly" is the Abstract one. Notice also another contradiction of Fraser's claim (see Appendix) that a nominal like "assembly" here is a Substantive. We can have "the plane's tail assembly fell off" (Substantive sense of "assembly"), which is very different from the kind of assembly intended in "automobile assembly plant".

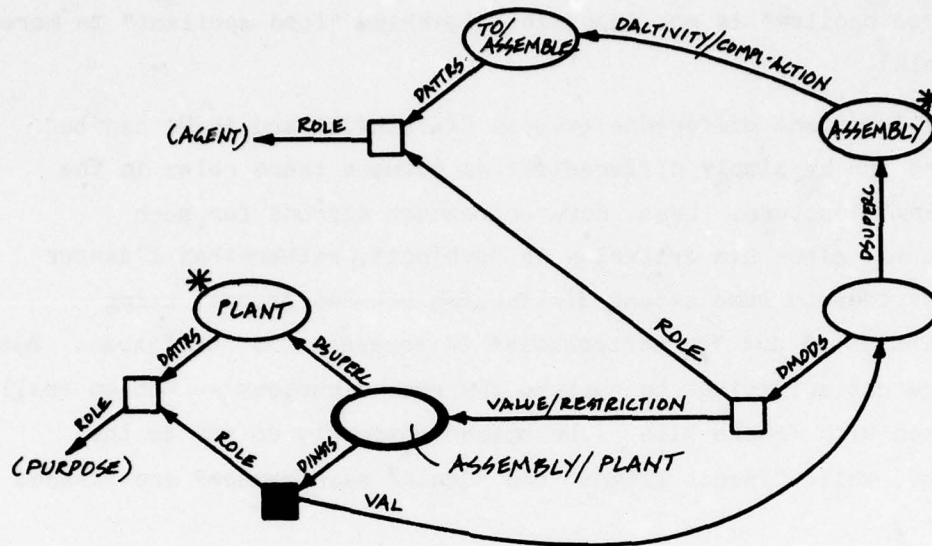


Figure 6.18. A V-S compound with "purpose adverbial".

present. The first of these is not generally considered a nominal compound.

6.4.1.4. Verb-Prepositional Object compounds

The other group of compounds which explicitly contain verbs is labelled by Lees the "Verb-Prepositional Object" class. This is quite an extensive group of compounds, since prepositions abbreviate a very large set of relations with a small group of syntactic function words. The group exhibits all forms of underlying compound structure that we have encountered so far:

	V-PO	PO-V
Infinitive	dance hall	-----
Activity PROCESS	washing machine	ocean fishing
COMPL-ACTION	recovery time	steam distillation (Abstract) boat ride (Generic)

Agentive ----- star gazer

While the relations between components here vary widely (LOCATION, MEANS, INSTRUMENT, TIME, and GOAL of the verb, to name a few), we can easily account for the compound structures with the mechanisms introduced above. In all cases, the nominal is derived from the verbal concept, and the appropriate dattr is modified (the modification can be applied to any of the dattr of the verbal concept). Then, in the cases where the prepositional object is the head of the construction, the restricted nominal is used to fill a role like PURPOSE, producing as the definition of the compound a restricted version of the head concept. There are roles other than PURPOSE which can be filled in some cases (as in compounds like "boiling point", "flying time", and "winning streak", etc.), but the basic structure is always the same. Figures 6.19 and 6.20 summarize the conceptual structure of this class.

At this point, it should be glaringly obvious that there is no significant difference among any of the three classes of compounds that I have so far discussed. OBJECT and "subject" (AGENT or EXPERIENCER) are roles closely associated with verbal concepts just as are the "prepositional object" roles like LOCATION, TIME, GOAL, etc. It is only the linear surface structure of English, which cedes to only two positions (those immediately preceding and immediately following the verb) the privilege of appearing without a preposition, that separates the former two from the others. Witness the fact that nominalizations easily add prepositions to even the special roles of AGENT and OBJECT, as in the "the fooling of the umpire by the shortstop" (from "the shortstop fooled the umpire") -- the general paradigm, in the end, should not have to differentiate between concepts whose surface manifestations differ in this way*. Compounds can be generated between

* In addition, captured in general is the underlying sense of the verb -- "boat riding" refers to the process of the travelling, while a "boat ride" speaks of the entire trip. Once again, Lees has trouble accounting for these different senses with his syntactically-oriented

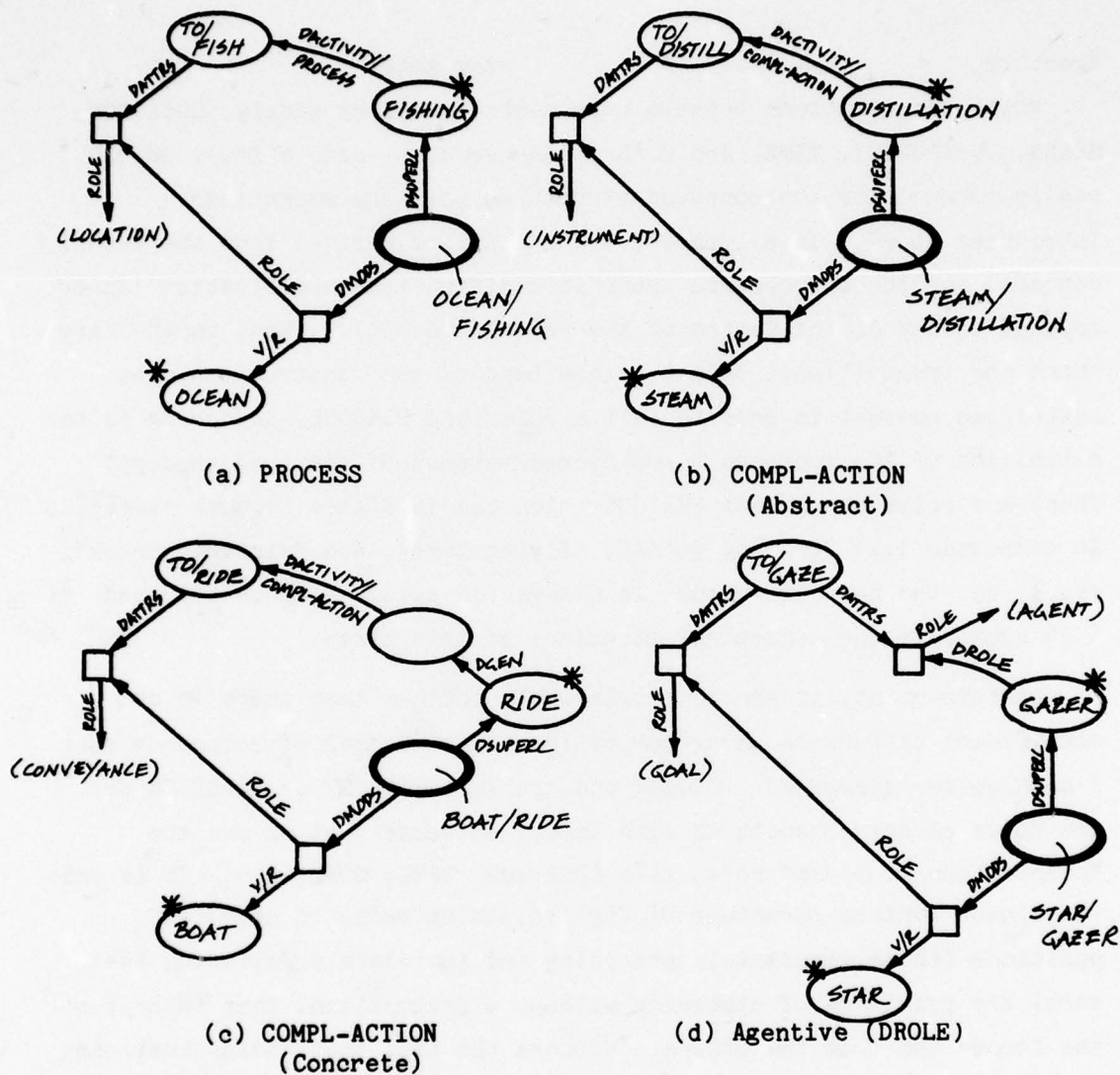


Figure 6.19. Nominalized verbs restricted by prepositional objects.

nominalized verbals and associated dattr with regularity, and, with either of these as the head of the pair.

nominals: "washing" in "washing machine" is a Gerundive, yet "recovery" in "recovery time" is an Action nominal.

Section 6.4.2
Noun-plus-dattr compounds

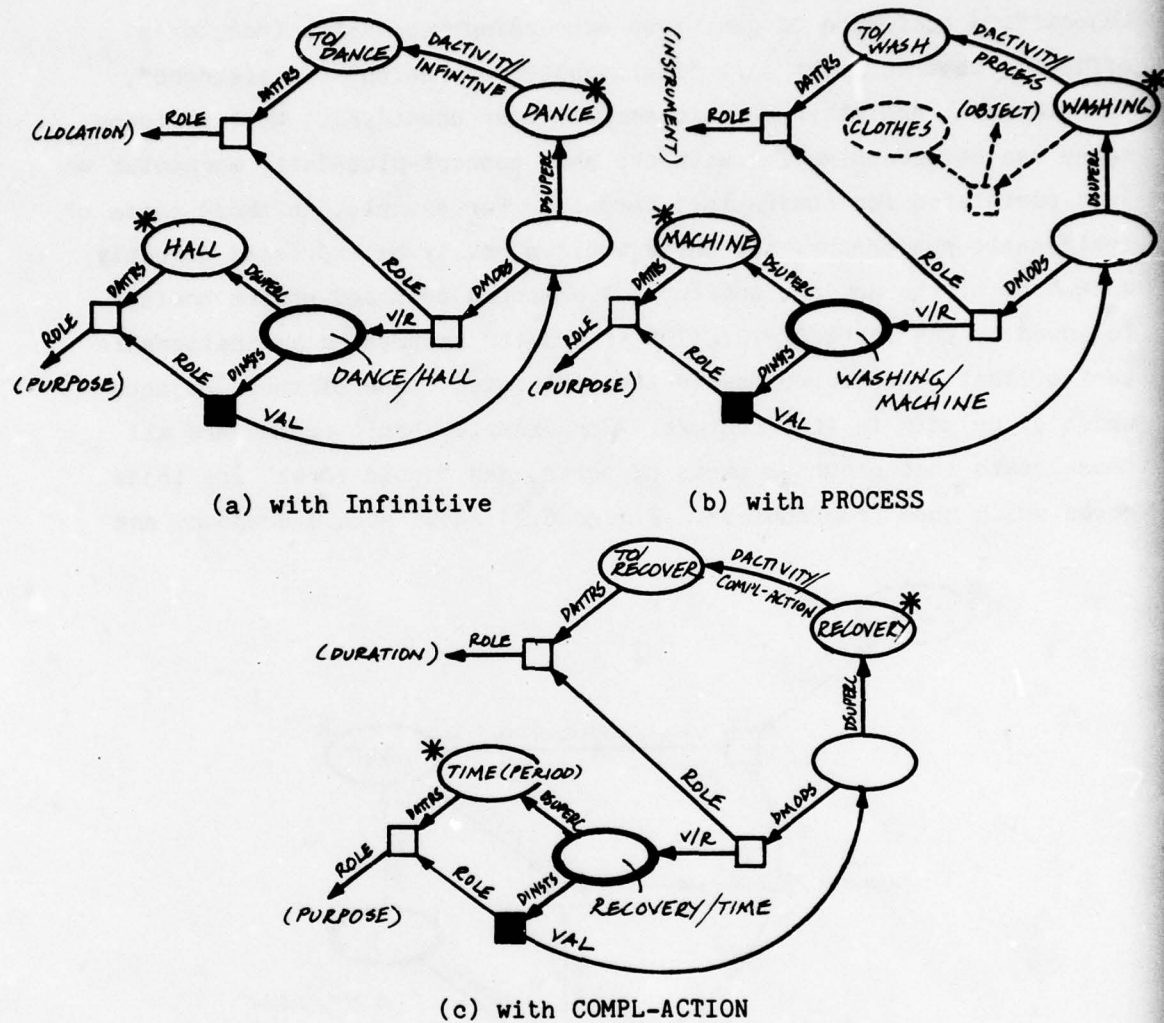


Figure 6.20. Prepositional objects restricted by verbal modifiers.

6.4.2. Noun-plus-dattr compounds

In fact, it is no special property of a verb to form such compounds, for as we have insisted, the idea of a dattr makes sense when associated with a strictly nominal concept (and we have seen how nicely this notion fits with the noun-like versions of verbs called "nominalizations"). Lees presents a class of compounds that he calls "(II) Subject-'Middle

Object'", a confusion of genitives expressing possession ("doctor's office"), what we might call "inalienable possession" ("apple core", "arrowhead"), and other relationships ("bear country"). Most of these cases can be accounted for with the same concept-plus-dattr mechanism we just postulated for nominalized verbals. For example, in those cases of inalienable possession, the parts would normally be expressed directly as dattr of the nominal concept. A compound composed of the nominal followed by the VALUE/RESTRICTION of a dattr expressing an inalienable part of that nominal represents the restricted class of those objects which occur only in that context. For example, "suit coats" are all those coats that occur as parts of suits, and "apple cores" are those cores which come from apples*. Figure 6.21 shows such a compound and

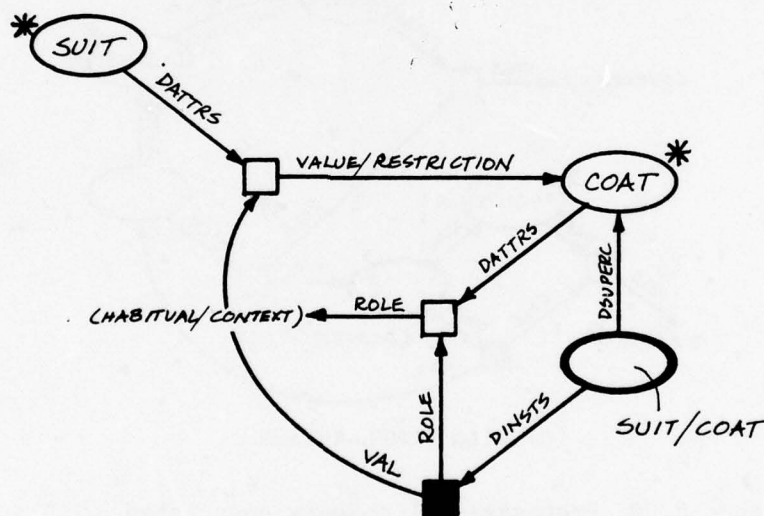


Figure 6.21. Nominal plus part.

its structure**. Notice the similarity of this construction with the

* Notice that the roles here can be used in compounds, as in "arrowhead" and "desk top".

** I am not satisfied with this analysis. Rather than have a peculiar role like "HABITUAL/CONTEXT", I would advocate moving the relationship to SUIT into the S/C of SUIT/COAT. This, in retrospect, would probably also better account for the "EATING/APPLE" example.

parts of those above that include the nominalized verbal, its subconcept, and the object. Also, notice the "habitual association" necessary to make a coat into a suit coat -- this same notion can help explain compounds like "doctor's office", to which we return in a moment.

The reversed compounds of inalienable possession seem to rely on an element of optionality on the part of the modifier. For instance, an "arm chair" is a chair that has arms; not all chairs do, so it makes sense to subdivide the entire class by this important discriminating feature. On the other hand, it is fine to say "desk top" to focus on a particular kind of top, but to say "top desk" is of no use, since all desks have tops. The MODALITY link can be used to block TOP/DESK, and allow ARM/CHAIR, as illustrated in Fig. 6.22.

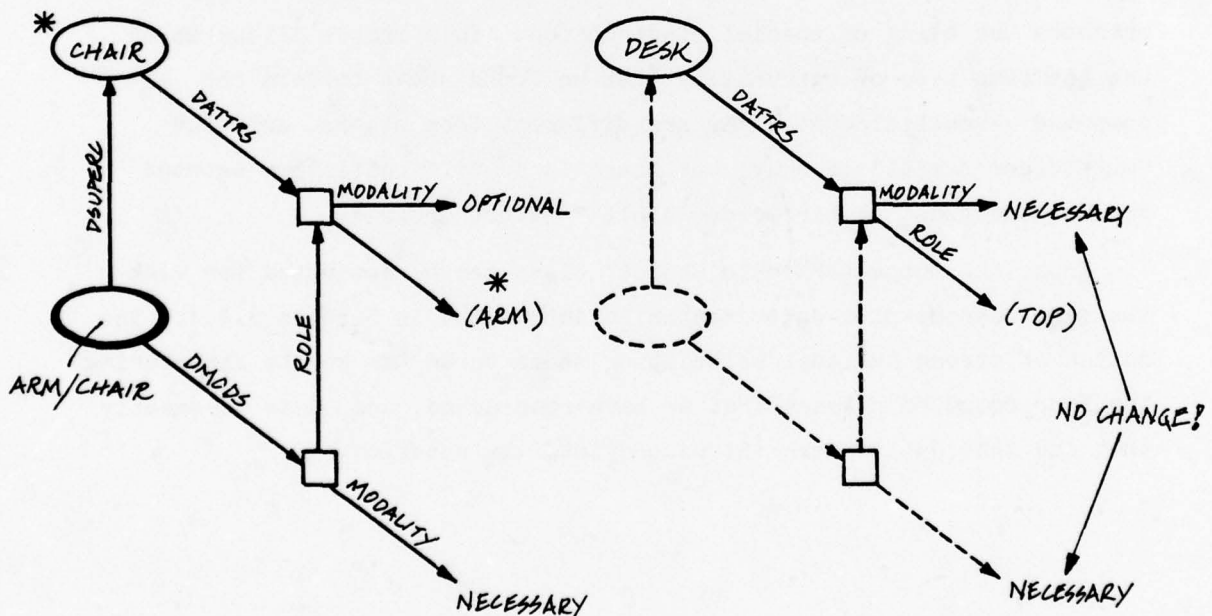


Figure 6.22. Optionality in choice of modifier.

"Bear country" and "Indian territory" have underlying structure identical to ARM/CHAIR, with an INHABITANT role filler doing the modifying. While the inhabitants of a certain country or territory may

not be considered strictly to be a "part" of that territory (in the same way that its geographical features and boundaries might be), the compounds that it can form are identical to those like "arm chair". Thus, it is not necessary to be strictly a part to form this kind of compound -- the definition of a dattr allows closely associated attributes like INHABITANT equal status with physical parts.

Those other compounds with possessives like "catcher's mitt" and "doctor's office" can be accounted for in a similar manner. In our own internal definitions of catchers and doctors we no doubt have some pretty strongly associated items like their mitts and offices, since these are things and places encountered very often, and which have special natures attributable to their association with catchers and doctors. Thus dattr for both CATCHER and MITT might plausibly refer to each other. All baseball players have mitts, the particular types that catchers use being of special construction. This status allows us to use the same type of optionality that we found above to form the compound -- outfielders' mitts are different from others, and thus "outfielder's mitt" is okay; but there is no differentiation between fields, and thus "centerfielder's mitt" is not useful*.

Thus, the Subject-"Middle Object" class can be accounted for with the same concept-plus-dattr mechanism introduced in Section 6.4.1. The notion of strong habitual association seems to be the key to structuring the four compound classes that we have considered, and it is to exactly that end that dattr were introduced into the notation.

* Why the possessive is used is not clear, except perhaps to emphasize the possessive, as opposed to inalienable part, relationship. However, terms like "goalie stick" are in common use.

6.4.3. Dattr-dattr compounds

Given our new understanding of the underlying structure of Lees' compound groups II, III, V, and VII, the relations upon which the other groups are based become clearer. Where all of our compounds so far were combinations of concept-plus-dattr, a compound like "car thief" can be seen to marry two different dattrs of some unspecified concept. Exactly what that concept is is a function of the particular knowledge existing in the knowledge base -- as I have stated, compounds with no verbal concept stated have many possible interpretations, as the "lion house" example of Gleitman and Gleitman illustrates (see Section 3.3.1).

This problem casts some doubt on the utility of Lees' breaking this group into "subject-object", "subject-prepositional object", and "object-prepositional object" categories. Consider his "subject-object" compound, "field mouse". If our underlying definition is based on the verbal concept, TO/INHABIT, then the subject-object interpretation is reasonable. But if we believe that field mice come from fields, the relationship is more like subject-prepositional object (in fact, the TO/LIVE/IN alternative to TO/INHABIT seems funny taking FIELD as an OBJECT). Or, consider "knife wound" -- surely, a KNIFE causes such a WOUND, but generally the knife is considered an INSTRUMENT, used by some animate agent to inflict the hurt. Thus, while "knife" may appear as surface subject in one manifestation, it is not the AGENT of the underlying conceptualization, and the subject-object categorization does not help us to understand the concept any better.

The key thing to observe here is that while many different surface manifestations might be possible to explain a compound, the underlying representation for it is what counts. The "lion house" example shows us that usually the many alternative descriptions paint basically the same picture -- in this case, the idea is of a house in which lions spend some of their time. To provide adequately this type of non-rigid definition facility (i.e., one that can vary depending on which concepts

are in a particular knowledge base), a representation must allow idiosyncratic definitions while preserving clearly the notion of role. While LION/HOUSE may have a complex, not completely specified definition, its representation must have a clear and unambiguous place for the two elements LION and HOUSE.

As we saw in Chapter 5, the explicit structural condition provides this facility. The particular HYDROGEN/BOMB example (Fig. 5.1) was just one of many possible ways to interpret how "hydrogen" and "bomb" might fit together. A most basic alternative might be as in Fig. 6.23, where

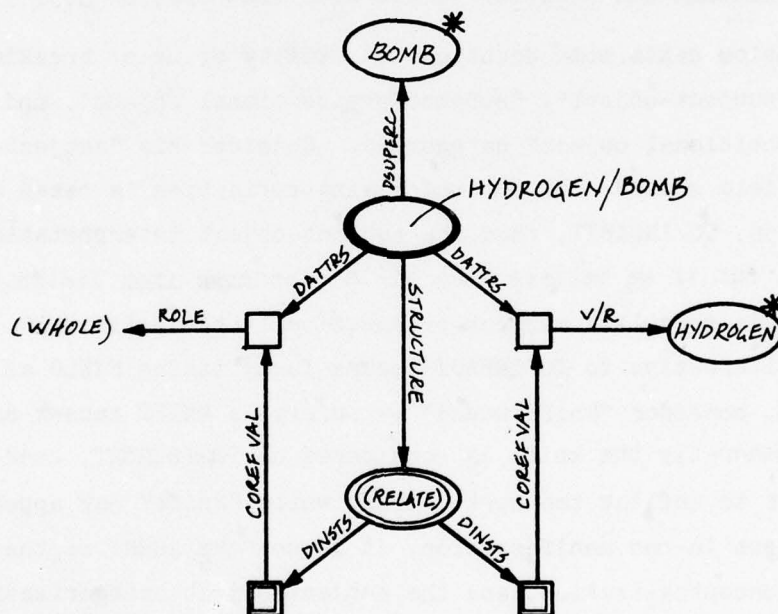


Figure 6.23. A very vague definition.

a minimal structural condition indicates that all we know is that HYDROGEN has something to do with the BOMB. A more detailed interpretation might use another concept like TO/POWER, as in Fig. 6.24. This conceptual structure indicates that a bit more of the relationship between HYDROGEN and BOMB is known; but notice that if TO/POWER were defined as vaguely as HYDROGEN/BOMB was in Fig. 6.23, the additional explanation would be only a very superficial one. Even in the most

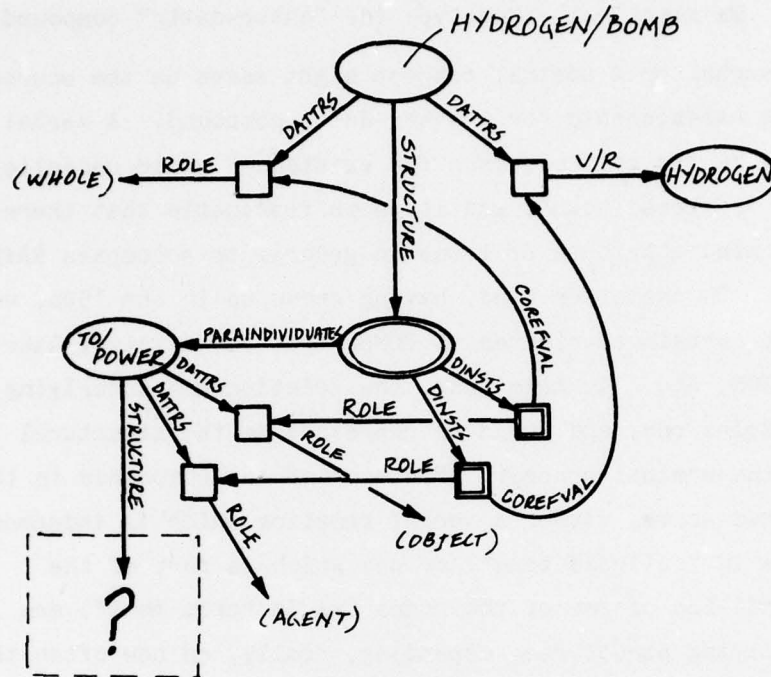


Figure 6.24. HYDROGEN POWERS a HYDROGEN/BOMB.

complete HYDROGEN/BOMB example (Fig. 5.1), the detail of potential explanation is dependent on the depth of definition of the concepts involved in the structural condition. In fact, as in all representations of this kind, such definitions are ultimately 1) non-defined (as above), 2) circular, or 3) primitive (i.e., defined in terms of some non-introspectable routines). Since we cannot obtain "complete" definition (we must eventually count on agreement of primitives between participants in a dialogue), differences in level of explanation here are just a matter of degree. Uniform SI-Net notation provides the right kind of explanatory capability -- it allows for a complete spectrum of "half-baked" ideas.

All of the remaining categories of Subject-Object, Subject-Prepositional Object, and Object-Prepositional Object can be accounted for as above, with conceptual structures in which the compound elements

are both dattrs of some (potentially complex) non-specified relationship. We might call this type the "dattr-dattr" compound.

Either a verbal or a nominal concept might serve as the source of the connecting relationship for a dattr-dattr compound. A verbal relationship like "to be the reason for existence" could underlie a compound like "railroad town", and it seems reasonable that there is no closely associated attribute of towns in general to encompass RAILROAD in this sense. On the other hand, having grown up in the '50s, we have come to expect certain attributes of BOMBS that cover things like ATOM, HYDROGEN, FUSION, etc. In this case, the relationship underlying "atom bomb" is a complex one, and would be expressed in the structural condition of the nominal concept. For many of the compounds in the groups mentioned above, either a verbal relation which is independent of both nouns (as in "railroad town") or one which is part of the structural condition of one of the nouns (as in "atom bomb") are possible underlying structures, depending, really, on how often the concepts that serve as heads are seen to be differentiated along the dimensions specified by the modifiers.

It would seem that new compounds can be formed either by differentiating a nominal along some already associated dimension (e.g., "bar", normally always associated with alcoholic beverages, becomes differentiated by what it serves, yielding "milk bar"), or by placing together two nouns with the head of the pair not having an aspect normally associated with the modifier (e.g., the business of the sponsor of an opera is not something normally associated with the opera, thus forcing a new relationship between SOAP and OPERA in SOAP/OPERA). Constant use would seem to force the latter type of relationship into the structural condition of the head noun.

6.5. Understanding compounds

The contrast between the treatment given to concept-plus-dattr compounds and the immediately preceding analysis of dattr-dattr compounds should serve to underline the difficulty involved in processing the latter type. When the verb is explicitly present in a compound pair, the task facing an understanding program is basically one of determining which role associated with the verbal concept is most appropriate for the other element. All dattr's of the verbal concept (including inherited ones) must be checked to isolate VALUE/RESTRICTIONS that cover the non-verb component. Those that are found (if more than one) then must survive a check of the structural condition in order to be considered valid candidates for the particular relationship between the parts of the compound. To complete the analysis of the compound, the proper type of nominalization for the verbal element must be determined, and the particular node for the compound itself must be constructed, depending on which element is the head (if the non-verbal element is the head, the internal connection to the restricted nominal must be made, using the "habitual action" notion introduced in Section 6.4.1.2)*.

For example, consider the O-V compound, "child rearing". The verbal concept TO/REAR would have dattr's for at least an AGENT and an OBJECT and a TIME (these might be inherited from a more general concept like TO/GROW, if such were to exist in our network). It is not unreasonable to assume that their VALUE/RESTRICTIONS might be PERSON, YOUNG/LIVING/THING, and TIME/SPAN, respectively -- see Fig. 6.25. In this case, a check on the VALUE/RESTRICTIONS of these dattr's of TO/REAR would reveal two viable candidates for the role that CHILD can play:

* I am describing the case of understanding a novel compound. The process of understanding a compound that is already known and stored in the network is of course much easier.

BBN Report No. 3605
Bolt Beranek and Newman Inc.

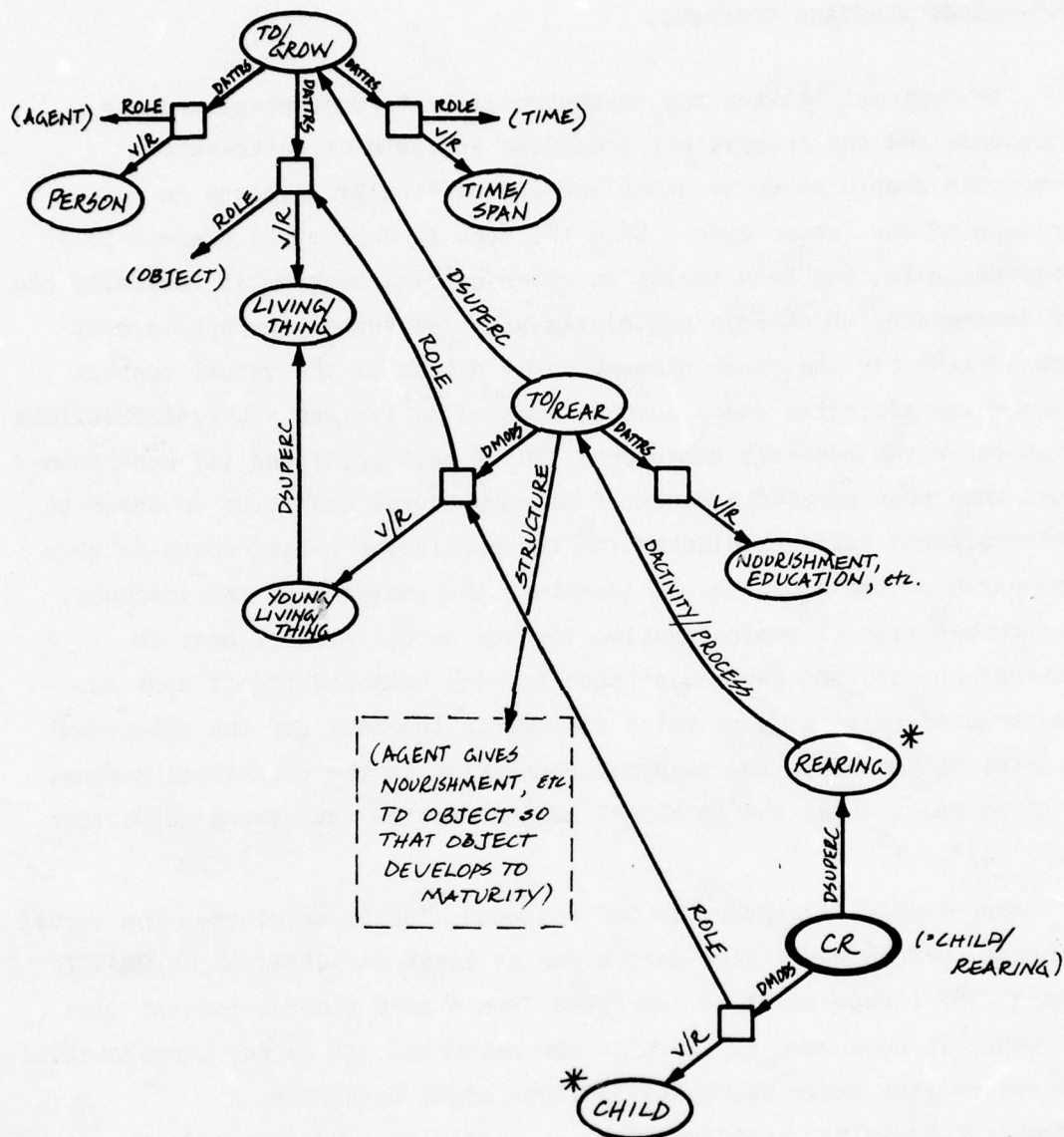


Figure 6.25. Derivation of CHILD/REARING.

AGENT and OBJECT. Since it is not possible to isolate the proper role using only the VALUE/RESTRICTION check, the modifying concept, CHILD, must be checked out as a filler of each of the candidates by substituting it for those roles in the structural condition. The structural condition of TO/REAR will express how the AGENT gives

nourishment, education, etc. to the OBJECT, so that it can develop to maturity. Substituting CHILD into the AGENT, then the OBJECT slot, we see that the compound could still conceivably be constructed using either dattr -- the CHILD could do the rearing (of a plant, or a pet, etc.), or it could be the thing reared. However, since the structural condition states that the OBJECT is brought to maturity, the fact that the definition of a CHILD involves its immaturity makes the OBJECT role a better candidate*. Finally, the syntactic context would presumably allow the selection of the Abstract version of the DACTIVITY/PROCESS nominal (in any case, TO/REAR does not seem to have a Generic version), and the node for the compound could be formed as in Fig. 6.25. Node CR is a subconcept of REARING, and since the OBJECT role of TO/REAR was chosen, it is restricted in the standard way, using the DMODS and VALUE/RESTRICTION links.

Notice that this analysis can be made even though the particular compound concept does not exist explicitly in the knowledge base prior to the perception of the two juxtaposed elements. As I have insisted, the syntax of the underlying network formalism determines in advance the shape of future concepts. The set of particular existing concepts,

* As I mentioned, CHILD fits the VALUE/RESTRICTION of both the AGENT and the OBJECT. However, the thing that differentiates CHILD from its superconcept, PERSON, is that it is immature. This would be specified by an instantiated dattr on CHILD. Since dattrs are inalienable attributes of concepts, they are in a sense "closer" to the concepts than are their DSUPERCs. This criterion would allow a program to choose between candidate dattrs based on descriptions in the structural condition.

Further, it would make sense in many concepts to express preferences in the structural condition -- in this case, we would want to express the connotation that the AGENT is usually an adult. This also would allow choices to be made between candidate roles.

Finally, it may be the case that certain verbs prefer certain roles for compounding. This preference probably arises out of common usage, and might be reasonable to express with the verb. Here, TO/REAR distinctly favors the OBJECT, at least in compounds with the nominalized verb as head.

along with the rules for role modification and instantiation, yield a large lattice of "implied concepts" that can potentially be understood in terms of existing ones*. This is where a well-conceived notation that breaks concepts into the right pieces can buy a lot of inferential power. Given, for example, a basic definition of "mechanic" (as, say, the Agentive nominal of TO/REPAIR/A/MECHANICAL/DEVICE -- see Fig. 6.11) which includes closely associated roles for the thing repaired and place of repair, we can automatically understand "airplane mechanic", "car mechanic", "foreign car mechanic", "VW mechanic", "Karmann Ghia mechanic", "shop mechanic", "garage mechanic", "home mechanic", etc., etc.

On the other hand, the lack of a verb upon which to focus presents a much more difficult problem for understanding the dattr-dattr compounds. In fact, it is not clear whether we can at this point offer any general methods for inferring the relation underlying a compound with two "pure" nouns. The structure of our concepts does, however, at least provide some clues to how a system might make "educated guesses" as to the relation between two nouns where none is indicated in the compound itself.

Let us briefly consider the compound, "hockey stick". If a system has a reasonably extensive description of the game of hockey, then it probably "knows", at some level, that it is played with a puck and sticks (notice that the particular manner in which it is played is not relevant to this compound, whereas to "hockey offsides" it most assuredly is). Therefore it could guess that a reasonable relationship underlying "hockey stick" might be formed between HOCKEY and a subclass of fillers of its EQUIPMENT dattr (the subclass being STICK).

* This implies, as has been my assumption all along, that the system must know at least general definitions in advance. This is not an unfair basic premise, since we could not expect a system to determine the meaning of a compound if it did not know what the two elements were.

If, in a different case, there is relatively little extant knowledge about hockey, per se, a good guess can still be made if enough superconcept knowledge exists. If it were known only that HOCKEY was a SPORT, and that SPORTs very often had EQUIPMENT, this dattr would be a prime candidate for a "reasonable relation" between the sport and the stick. Presumably, there are very few other aspects of SPORTs whose VALUE/RESTRICTIONS would allow STICK as a role filler (perhaps something associated with a goal or physical plant might be a candidate -- one could conceivably guess a HOCKEY/STICK to be like a HORSESHOE/STAKE). Figure 6.26 illustrates the derivations of the preceding two cases.

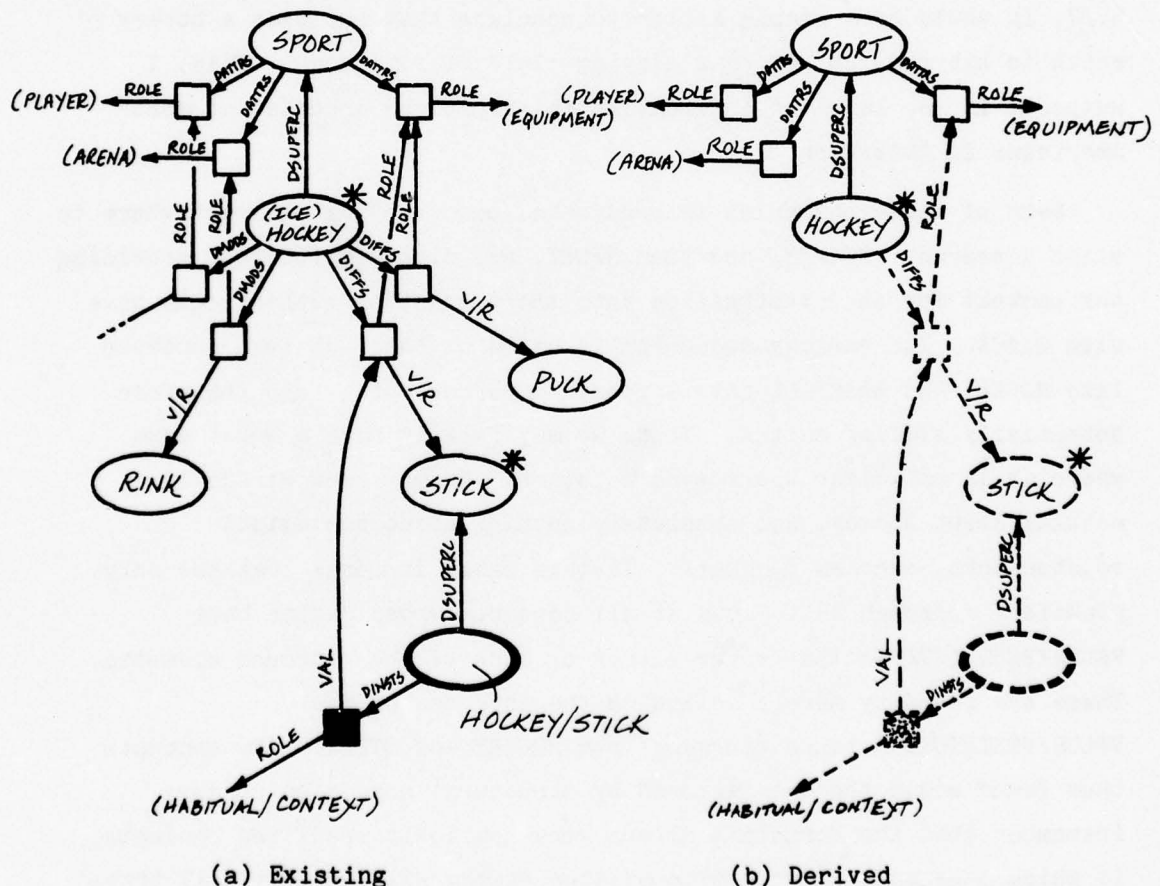


Figure 6.26. An existing compound and a derived one.

6.5.1. Analogies

Consider still that our program might know very little about hockey-like games or hockey itself, but that it might have heard of a "baseball bat". Given the similarity between BAT and STICK, the very naive knowledge that HOCKEY is like BASEBALL (i.e., that they have the same superconcept) would be sufficient to attempt an explanation by analogy. Therefore, given the conceptual structure illustrated in Fig. 6.27, it would be a simple matter to conclude that one uses a hockey stick to hit some object on a playing field of some sort. This, I suspect, is the level of inferential ability to be expected of most Americans in this case.

Each of these scenarios is predicated upon knowing at least where to start a search. HOCKEY, and then SPORT, are directly located, providing the context for an investigation into the connection either might have with STICK. The analogy mechanism is based on the fact that concepts like HOCKEY and BASEBALL have a common superconcept -- and therefore potentially similar dattr's. Thus, we may finally have a worst case where it is not clear where even to start. We may know virtually nothing about hockey, and absolutely nothing about analogical counterparts, such as baseball. In this case, it seems that the only plausible approach is to look at all concepts whose dattr's have VALUE/RESTRICTIONS that cover either or both of the compound elements. These are found by merely following the inverses of the VALUE/RESTRICTION links emerging from HOCKEY and STICK. Any concepts thus found would then be filtered by structural condition checks (remember that the formalism allows very shallowly specified concepts, in which case many structural condition checks will be trivially true, and the ultimate conclusion may be as vacuous as "something used in hockey").

Section 6.5.1
Analogies

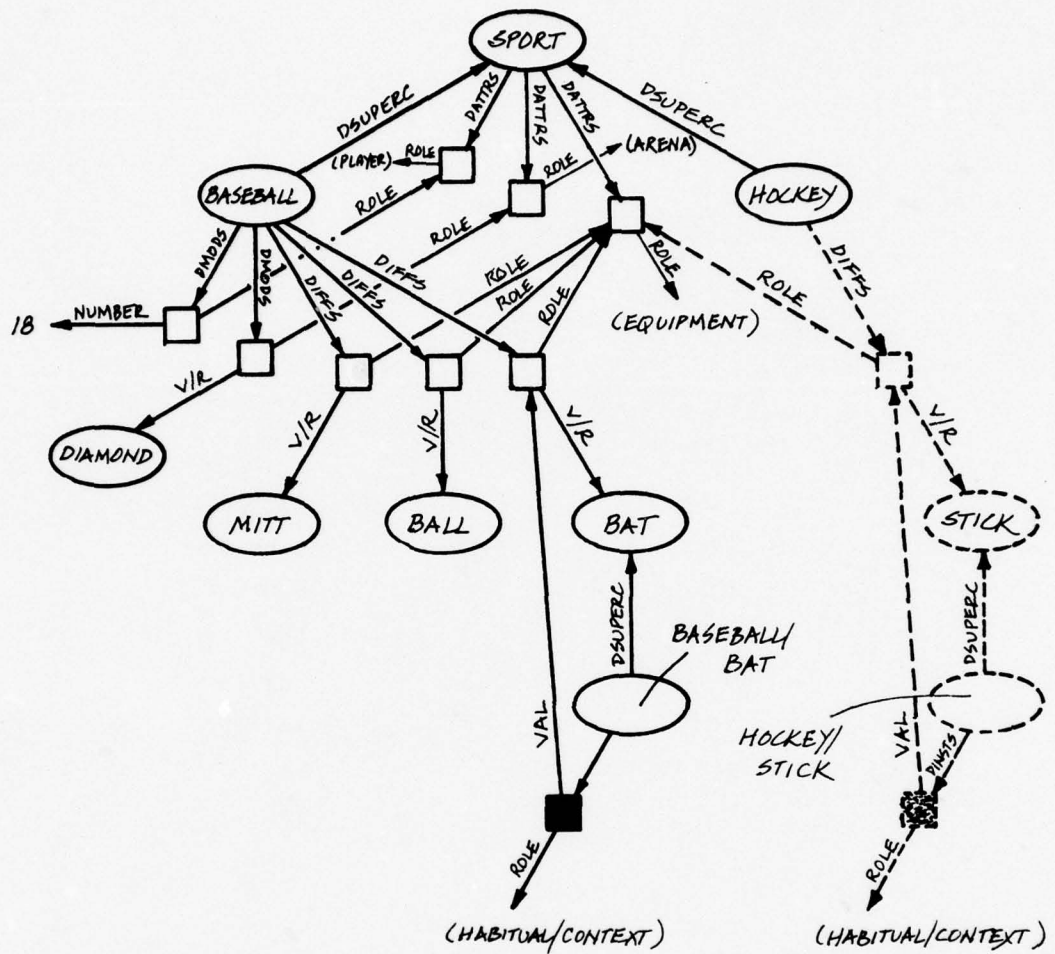


Figure 6.27. HOCKEY : STICK :: BASEBALL : BAT.

BBN Report No. 3605
Bolt Beranek and Newman Inc.



Chapter 7. Representing Knowledge about Hermes

Introducing a new user to a large and sophisticated computer program is a difficult task. Even if the program is engineered to appear simple, the user will often need to refresh his knowledge of command structure and function, and to ask questions about useful features of which he has not yet taken advantage. A useful application of knowledge representation would thus be the construction of an intelligent on-line assistant which "understood" the target system, could simulate its mechanisms, and could answer questions about both the program's operation and hypothetical situations.

One basic requirement for such an intelligent agent, as discussed in Chapter 3, is a thorough and accurate description of the target program. Such a description must provide access not only to the objects and commands themselves, but to how the objects are structured, how the commands use the objects, and how the objects may be conceptualized by the user. The program should be able to use explicit definitions and descriptions to understand paraphrase queries, so that questions phrased in many different ways might be answerable. Also, the assisting program should be able to generate explanations and alternative descriptions from the knowledge base itself.

This chapter investigates the appropriateness of the SI-Net structural paradigm, in particular its emphasis on definitional connection, for representing knowledge about the Hermes message-processing system [Myer, Mooers & Stevens 1977]. Here I will make extensive use of the important new features of SI-Net representation, and exemplify how a complete description of Hermes in this formalism would be possible.

7.1. -- Internal structures for Hermes objects

A great deal of the message-manipulating power of Hermes comes as a result of its maintenance of a set of objects. As I have mentioned, the user can create and save output templates which specify the format in which a message gets printed. He can also create filters, which are used to select actively particular subsets of messages from larger sets by specifying groups of desirable properties. Further, Hermes has a facility for saving explicit sets of messages as objects called sequences. There also exists a block of switches, which determine the defaults to be used in many situations. And of critical import is the message, the object Hermes sees when a new piece of text has been dropped in the user's "inbox". This incoming message has a closely-related counterpart -- the outgoing, or draft message. A draft is the kind of object a user creates to send to another ARPAnet user, and he can create and manipulate fields of the outgoing message in a very flexible manner.

One of the first things we notice about these objects is that they are not usefully thought of as indivisible entities. (Each kind of object has, in fact, a Hermes subprogram, or editor, available that allows the user to get inside the object and manipulate its parts.) For instance, to build a template, the user has to specify which fields of a message he wants output, in what order, and what, if any, interlaced text he wants printed (the very same templates can be used, by the way, to guide the prompted composition of a draft message). Since Hermes' interpretation of a template depends completely upon that template's internal composition, a Hermes on-line assistant must know about and understand that internal structure. This, then, is one of our principal representation tasks -- the representations of types of Hermes objects, including the definitions of their parts and how those parts go together.

Even though the most common kind of semantic net notation does not generally deal with the internal structure of nominals (objects), but rather is biased toward verbal concepts, we could imagine a straightforward extension of that notation to handle objects with known parts. For example, the Hermes "switches" are a block of a twenty-four alterable settings, each with a predefined allowable set of possible positions. We might try to represent the concept of such a switchblock as a node with a link for each switch, as in Fig. 7.1.

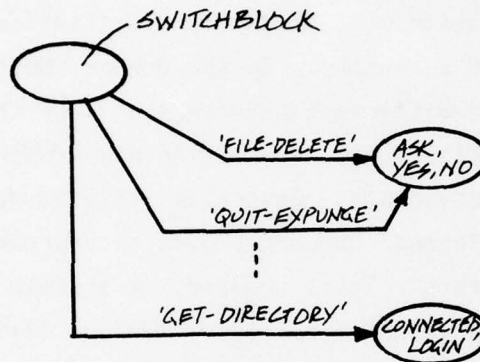


Figure 7.1. A non-viable attempt at SWITCHBLOCK.

As was pointed out in Chapter 4, such a notation is inadequate for several reasons, among them the way in which relationships like "FILE-DELETE" would be used ambiguously when an attempt to "instantiate" (individuate) SWITCHBLOCK was made, and the fact that many entities may fill the FILE-DELETE role cannot be specified. With other Hermes objects, the inadequacy of such a representation is even more telling. For templates, filters, and drafts do not have simple, fixed structure. A template is, first, composed of lines, the number of which is not predetermined. Each of those lines is in turn made up of items, again, of indeterminate number; the items, however, are not constrained like the values of the switches are -- they can be described in advance, but the set of potential values cannot be given. So, while perhaps any particular template could be represented by a structure like that of Fig. 7.1, such a representation cannot account for the important

properties of templates in general.

To handle the template case, an adequate notation must allow a description without trying to enumerate the class of potential fillers of the description. For instance, one of the items allowed in a template is a "literal". It is specified by the keyword "LITERAL", followed by an arbitrary quoted string. A notation must be able to capture the description of the class of strings that are legal, but cannot expect to do that by enumerating all of the members of the class. Finally, an adequate notation must allow the specification of the structural composition of an entity. In the common, unstratified, semantic net, there is no difference between the links that assert relationships between entities and those which might express the internal structure of individuals. Generally, this is gotten around by defining a fixed set of "cases" which are used to represent the internals of verbal concepts. This, however, is totally inadequate to handle nominals (and even verbals, really -- see our discussion of "cases" in Section 5.1.3.1).

The notation evolved in Chapters 4 and 5 is more suitable to this representation task. Dattr's are intended to define closely associated attributes, including the parts of nominal-type concepts, in a clear and unambiguous way. Also, the SI-Net structural representation is geared toward intensional description without bias toward existing objects, so that objects more complex than simple switches can be perspicuously defined. At first, an SI-Net notation for the concept of the Hermes SWITCHBLOCK would not appear much different from that of Fig. 7.1; such a representation is illustrated by Fig. 7.2. Remember that the concept node in this figure represents the concept of the SWITCHBLOCK, and schematically reflects what a single incarnation of a switchblock will look like (we will focus on the individuation mechanism in Section 7.4).

Hermes does change, however, and this concept of SWITCHBLOCK will have fewer or more dattr's depending on the particular Hermes system. Yet, through these changes, there is still a general notion of

Section 7.1
Internal structures for Hermes objects

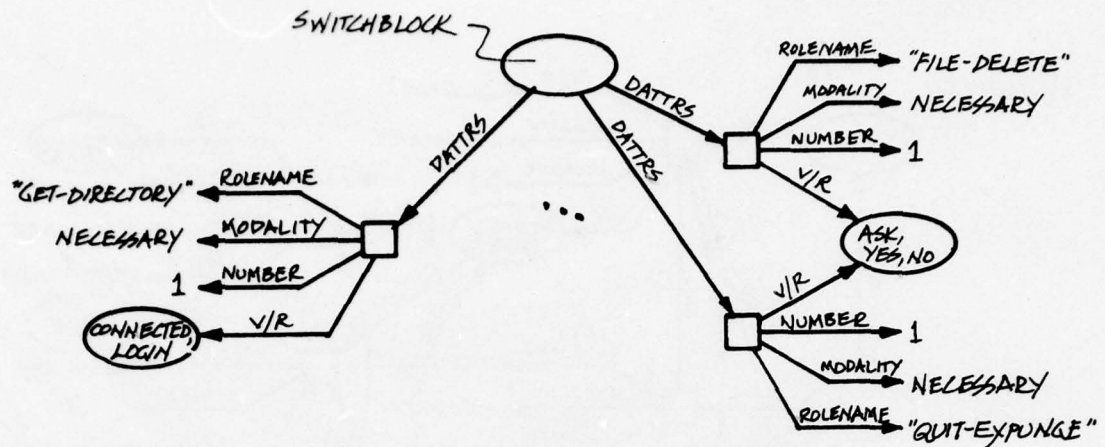


Figure 7.2. The dattr of SWITCHBLOCK.

SWITCHBLOCK that we would want to preserve, so that our intelligent assistant might be able to explain the concept independent of the particular switches in the system. So the switchblock might be better explained as in Fig. 7.3, with the particular switches being differentiations of the more general role of a SWITCH. Part of the definition of the SWITCH role is embodied in the definition of the SWITCH/STRUCT object type. The structural condition of that concept would express the fact that some relationship must exist between the SETTING of the switch and the COMMAND affected by that SETTING. Notice, then, that each particular SWITCH would have a modified structural condition that expressed the particular effect that that switch had on its own associated command. That is, the general concept of a SWITCH/STRUCT can, at best, have a structural condition that says "the SETTING affects the EFFECT (a dattr) of the COMMAND"; the FILE-DELETE SWITCH can further qualify this general condition by stating how each value (YES, NO, ASK) affects how the FILE command works. A particular user's switchblock will have the FILE-DELETE switch set to a particular value which will completely determine its effect. With a cascaded representation such as that illustrated in Fig. 7.3, an intelligent assistant program could follow well-defined links to determine all of the implications of a particular switch setting, even in the case of a proposed hypothetical switch.

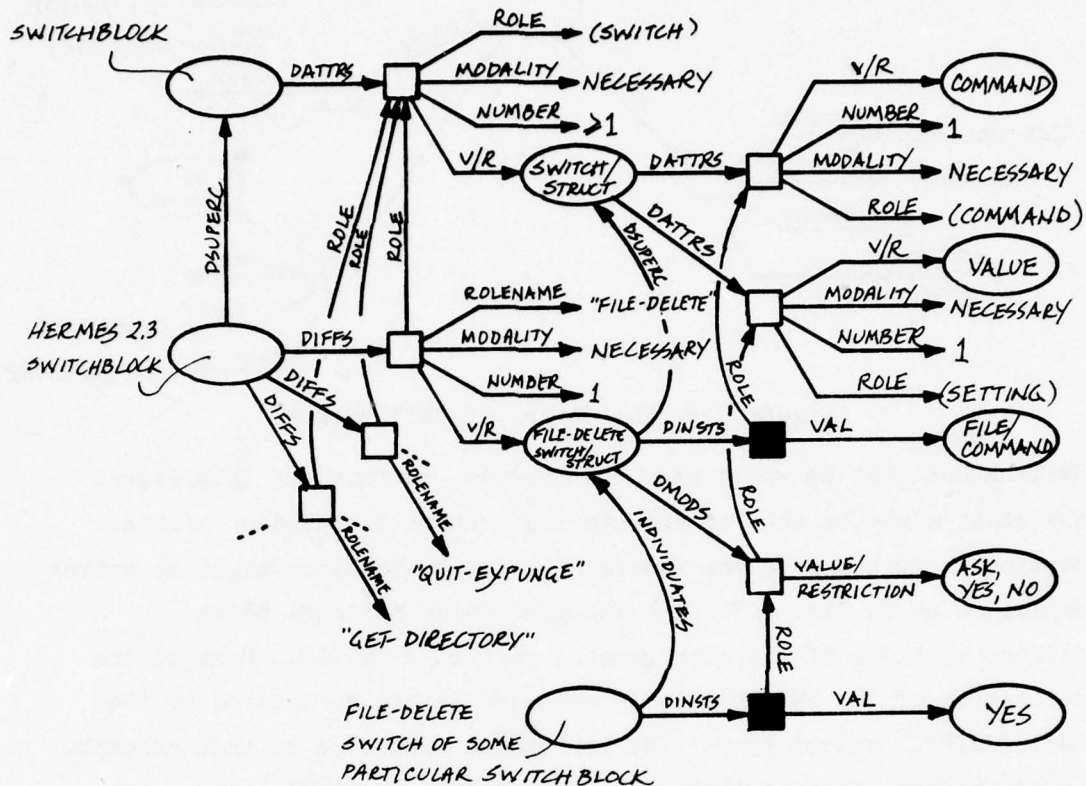


Figure 7.3. SWITCHBLOCK, in detail.

The multi-level representation is also appropriate for other Hermes objects. For example, as I have stated, a template is a series of lines, made up of items*. The top-level structure for the concept of a Hermes template will resemble that of the switchblock, but since there is no fixed format for templates, there is no need for an intermediate structure like "Hermes 2.3 template". Figure 7.4 illustrates a possible

* Template items are names of fields of messages, optionally followed by plus "+" signs; or, they are special items. One such special item is the "literal", which is followed by a quoted string. When the template is invoked for printing a message, the contents of each field of the particular message designated by an item in the template will be printed. If a plus sign appears, the name of the field will be printed in front of its contents.

Section 7.1
Internal structures for Hermes objects

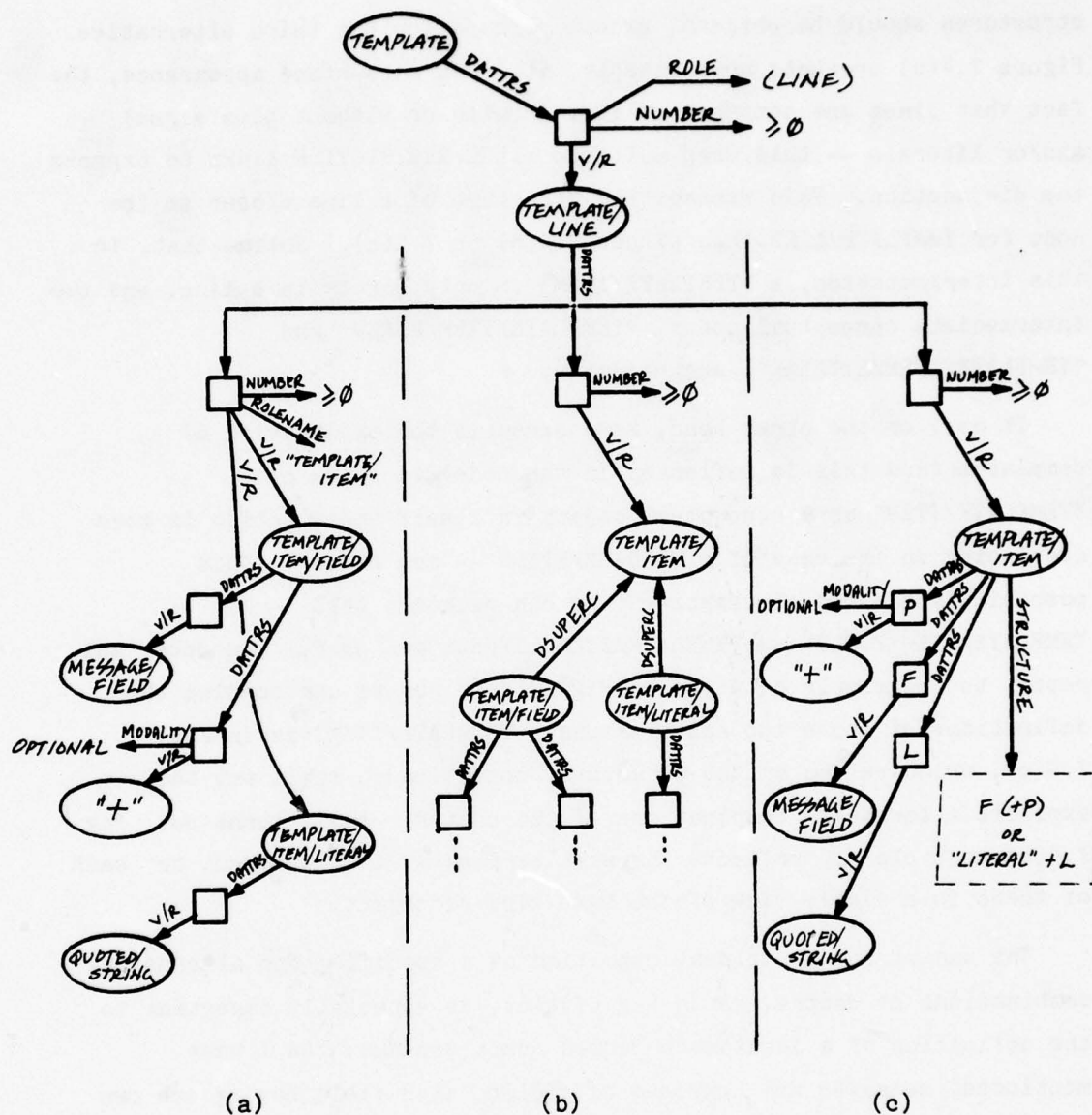


Figure 7.4. Structured substructures.

structure for templates, with three alternatives for the structure of a TEMPLATE/LINE. The alternatives are labelled "(a)", "(b)", and "(c)", and each should be considered to be attached by a DATTRS link to the node TEMPLATE/LINE. Each is to represent an alternative view of the structure of a template line. The interpretation of the three

structures should be obvious, except perhaps for the third alternative. Figure 7.4(a) presents more clearly, at least in surface appearance, the fact that lines are composed of fields (with or without plus signs) and/or literals -- this uses multiple VALUE/RESTRICTION links to express the disjunction. This exposes the structure of a line closer to the node for TEMPLATE/LINE than either 7.4(b) or 7.4(c). Notice that, in this interpretation, a "TEMPLATE/ITEM" is only a role in a line, and two intermediate conceptual nodes, "TEMPLATE/ITEM/FIELD" and "TEMPLATE/ITEM/LITERAL", are necessary.

It may, on the other hand, make sense in the explanation of templates (and this is reflected in the code) to think of a "TEMPLATE/ITEM" as a conceptual object in itself (this notion is more convincing in the case of a MESSAGE/FIELD -- see below). This possibility has two alternatives: we can maintain that TEMPLATE/ITEM/FIELD and TEMPLATE/ITEM/LITERAL are useful concepts, and resort to the simple notation of Fig. 7.4(b); or we can combine the definitions of those two entities under TEMPLATE/ITEM, as in Fig. 7.4(c), which relies on the structural condition to spell out the explicit alternative combinations of the dattr's. As it turns out, Fig. 7.4(c) most closely reflects the way the program is organized, but each of these is a viable view of the same kind of object.

The use of the structural condition as a specifier for alternative combinations of dattr's, as in Fig. 7.4(c), is especially important to the definition of a legitimate Hermes draft message. As I have mentioned, messages are composed of fields, each field having its own internal structure. For example, Fig. 7.5 shows how we might make use of the intermediate nodes "ADDRESS/FIELD" and "TEXT/FIELD" to define "TO/FIELD" and "SUBJECT/FIELD". At the most general level, the Hermes assistant could think of a MESSAGE as a collection of MESSAGE/FIELDS, just as a SWITCHBLOCK was a collection of SWITCHes. However, there are two different kinds of messages in Hermes, both of which it is necessary to know about. While both fit the very general characteristics of

Section 7.1

Internal structures for Hermes objects

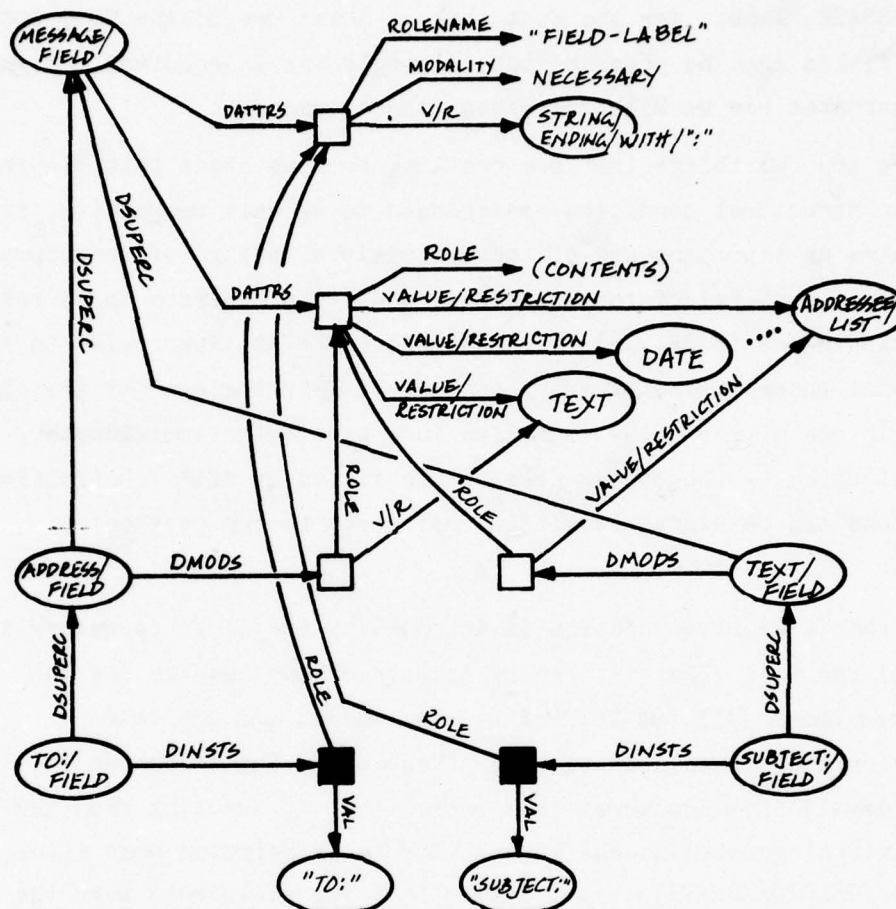


Figure 7.5. Some MESSAGE/FIELDS.

MESSAGES, draft fields each have closely associated commands for their creation (the command is simply the name of the field), and there are constraints on the presence of certain of the fields before a draft may be sent.

Thus, the structural condition for DRAFT/MESSAGE serves two purposes. First, it states, essentially, the "meanings" of each of the fields (for instance, that the members of the TO:, BCC:, and CC: fields are the intended recipients of the message); and second, it embodies the constraints on the presence of certain required fields. This latter would be adequately handled by the MODALITY links for the dattrs of

DRAFT/MESSAGE, except for the fact that at least one of the TO:, BCC:, and CC: fields must be present, but no single one is required. Figure 7.6 illustrates how we might represent these concepts.

There are two things that are critical to note about this figure. While the structural condition is intended to be only suggestive, it illustrates an important use of the definitional nature of structural conditions. Part (a) of the structural condition accesses three role description nodes (R, S, and T). What the COREFVAL links refer to are intensional descriptions of the potential fillers for each of the three roles. If one of the roles is filled in a particular individuator, the (OR) will apply -- the others need not be filled at all*. Definitional connections can be expressed without reference to any particular objects.

The same kind of connection is intended by the links to dattr in parts (b) and (c). That is, the recipients of the message are the addressees which fill the TO:/FIELD, etc. roles, and the role description node is an abbreviation of sorts for those unknown (at this general level) individuator. But notice that if the link from the SEND paraindividual pointed to the TO:/FIELD role description node (i.e., node TO:/FIELD/OF/DRAFT) it would seem that the recipients were the TO:/FIELD. On the other hand, if we pointed only to the CONTENTS dattr of TO:/FIELD, we would be indicating that DRAFT/MESSAGES were sent to all addressees of TO:/FIELDS (remember that incoming messages have them, too), not just the ones filling that field of the particular draft in question. Thus we need the two-headed FOCUS/SUBFOCUS pointer -- one head to pin down the context (i.e., this draft), the other to point out the relevant substructure of a substructure of the current concept. This turns out to be a common operation in Hermes -- most of the objects

* We probably need an explicit way to indicate the requirement that a dattr be filled. It is not clear whether a COREFVAL link to a role description node should work the way that is implied here.

Section 7.1
Internal structures for Hermes objects

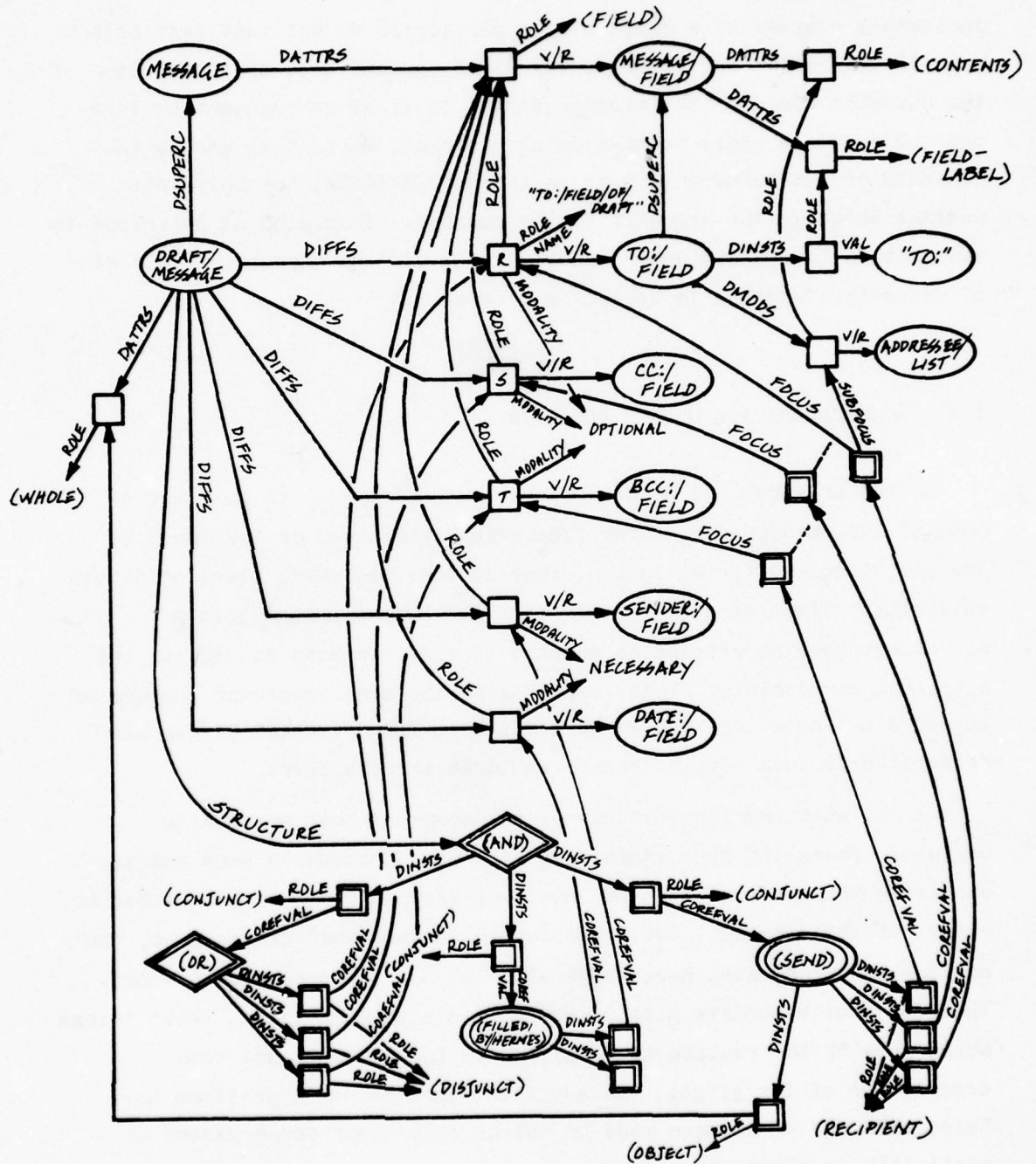


Figure 7.6. Draft message.

have parts which are themselves structured, and we need often to indicate a subpart of a subpart of a particular object (not that object type in general). For example, the REPLY command uses the first line of the SUBJECT: field of the message passed to it as an argument to form the subject of a reply message being created. We need to access the CONTENTS of a SUBJECT:/FIELD of an INCOMING/MESSAGE, but only that message which is the argument to the command. This kind of reference is made with a "composite dattr function"; I discuss the use of this kind of accessing function in detail in Section 7.3.

7.2. A hierarchy for Hermes commands

In the treatment of Hermes objects, we have begun to make use of some of the concept-derivation facilities introduced in Section 5.2. The use of role differentiation, restriction, and particularization can be further illustrated by representing the Hermes commands in a hierarchy, keeping attributes generic to many commands as high in the hierarchy as possible. This is much like the same important feature of standard net notation, but as we shall see, "inheritance" of general properties is less simple than it would at first appear.

First, what are the features of the commands that we need to capture? Among its more visible aspects are a command's name and its syntax -- this information the user must know in order to get Hermes to carry out the command. But, in order to "understand" the program, the on-line assistant must have information on how the code itself works. Thus, we must associate with a command its arguments (i.e., those things acted upon by the routine that implements the command) and some description of its effect. In addition, some Hermes operations have "side effects" -- changes made to things other than those passed in explicitly as arguments.

Section 7.2 A hierarchy for Hermes commands

Finally, we might consider the fact that an on-line assistant will have to deal with the way a user thinks about the program. A look at the documentation offered to teach new users about the Hermes system and a review of the kinds of questions asked about the system reveal that the concepts underlying the user's view of Hermes are not necessarily those embodied in the program itself. For example, people often make reference to "objects" that they believe Hermes creates, but which have no ontological status as far as the program is concerned. For example, asking Hermes to "make a listing" is very different from telling it that you want to create a template, since no Hermes object corresponding to the listing is created. Since the assistant should ultimately deal with questions from real people, it should be able to transform requests like "make a listing" to the Hermes command LIST. In addition, some notion of the function of each command should be included to handle the teleological view of Hermes that users have. Hermes' view of the world is quite limited and not teleological.

Figure 7.7 illustrates how these intuitions about commands might be mirrored in SI-Net notation (it should be emphasized that this is Hermes' notion of a command, and does not necessarily reflect the ordinary language notion). Note that the ARGUMENT role points to ARG/STRUCT, itself a structured concept. The use of a structured concept as VALUE/RESTRICTION, as illustrated in the previous section, is necessary here because, for each argument, there is a closely associated default value. This is the value that is used when no value for the argument is supplied by the user; and thus an ARG/STRUCT must have two parts -- one to represent what normally would be considered the argument (i.e., the VALUE), and one to specify the default. Note that the structural condition of this concept should indicate that the DEFAULT/VALUE must itself be a legal filler for the VALUE role. Also notice that while the parts of an implemented command might be limited to its arguments and effect (the routine body), SI-Nets allow us to define some closely associated things which are just as necessary to a

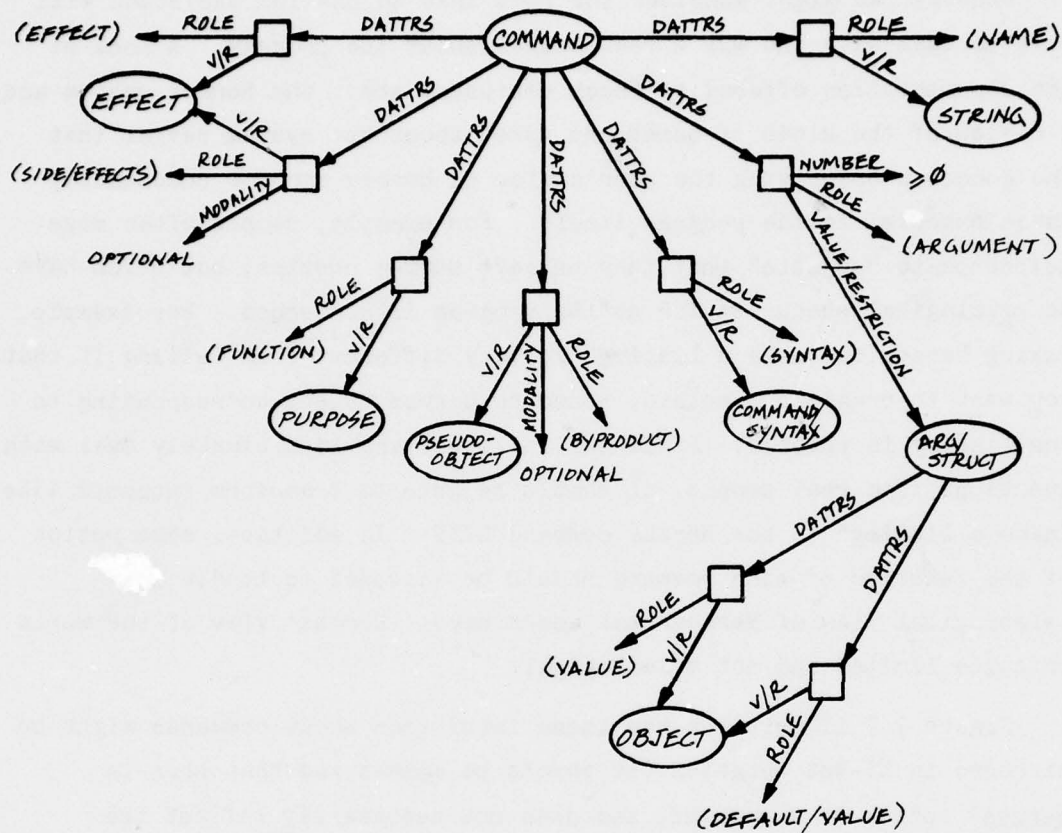


Figure 7.7. Hermes' notion of COMMAND.

complete understanding of the concept as those "parts" are. In fact, what the parts of a command are is a matter of how you look at it, and it is not necessary here to draw an arbitrary distinction between parts and non-parts. The relationship of each dattr to the others and to the whole should be clear from the structural condition.

The structural condition at the level of an undifferentiated Hermes command can only be very general -- all that might usefully be said here are some very general things about SYNTAX, ARGUMENTs, etc. At the node for the general definition of COMMAND, we can, for example, know that all particular commands have EFFECTs which will somehow manipulate their ARGUMENTs, but we cannot offer in advance what those effects might be. I will not attempt to specify that information here, except to point out

Section 7.2
A hierarchy for Hermes commands

that we do have a way in this notation to indicate "the objects passed in as arguments" intensionally -- that is, as a description of all those potential, but not yet specified, arguments to Hermes commands. To accomplish this, we can point to the VALUE role description node of the structured concept ARG/STRUCT, which is pointed to by the ARGUMENT role description node of COMMAND. I will return to this intensional kind of reference when I consider in more detail the connections between commands and objects in Section 7.3.

Now let's look at how the general node for COMMAND can spawn representations for some of the particular commands available to the system user. First, we might consider it useful to generalize from certain groups of commands some of their common properties, and refer to the command groups as, for example, "transcribing commands", "filing commands", "editing commands", etc. I will focus here on the set of Hermes operations for transcribing messages, and assume that the treatment applies equally well to the other command groups. In any case, rather than consider a single level hierarchy with all individual commands pointing directly to COMMAND, we will investigate a multiple-level structure which merges common properties.

The property that serves to separate the transcribing commands from all others is their uniform purpose -- they are all used to produce user-readable output of the contents of messages. We could also generalize the particular effects that these commands have on the Hermes objects -- they each use a template as a pattern for moving particular pieces of incoming (as opposed to draft) messages to output files. Each of these properties may be further restricted in smaller subgroups of commands; but they do serve to separate this major group from the others. If a user were to ask "how do I read my mail?" this would be the set of commands about which he should be informed.

The way that this grouping would be represented is illustrated in Fig. 7.8. A DSUPERC link indicates the subconcept-concept relationship between TRANSCRIBING/COMMAND and COMMAND. As mentioned, this particular

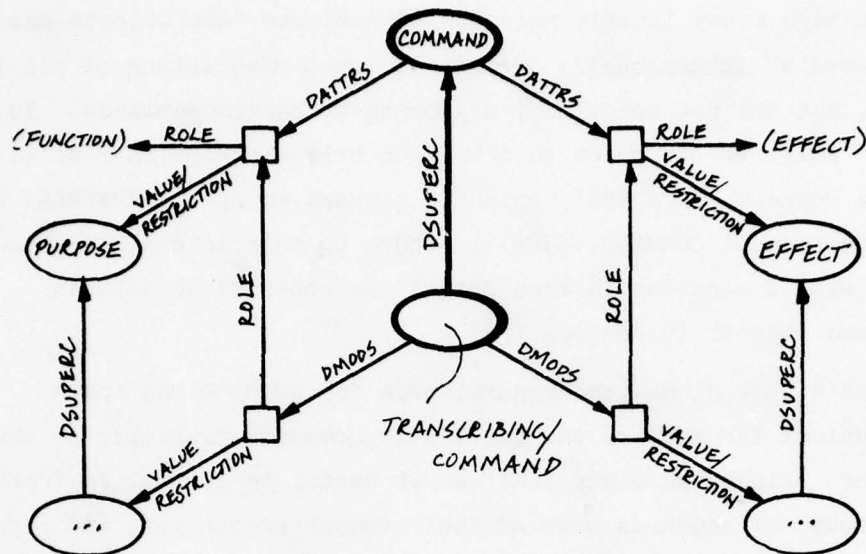


Figure 7.8. Transcribing commands, part 1.

subclassification serves to modify the FUNCTION and EFFECT dattr, and such modifications are indicated with DMODS links and appropriate modificational role nodes.

The next useful level of subclassification might be made along a different dimension. There are two commands which take no arguments (actually they are invoked by typing single characters -- <LINE-FEED> and <UPARROW>) and print only one message. All of the others, on the other hand, take at least two arguments, the first of which is always a message sequence, which is the group of messages ultimately transcribed by the command. Thus, we might represent SINGLE/MSG/-TRANSCRIBING/COMMAND and MULTIPLE/MSG/TRANSCRIBING/COMMAND as in Fig. 7.9 (heavier lines indicate the inter-concept links forming the basic hierarchy).

The five commands in the latter group separate naturally into two subclasses, the summarizing commands (SURVEY, SUMMARIZE), and the printing commands (PRINT, TRANSCRIBE, LIST). In the first subgroup, the two commands are identical except for the default sequence used when no

Section 7.2
A hierarchy for Hermes commands

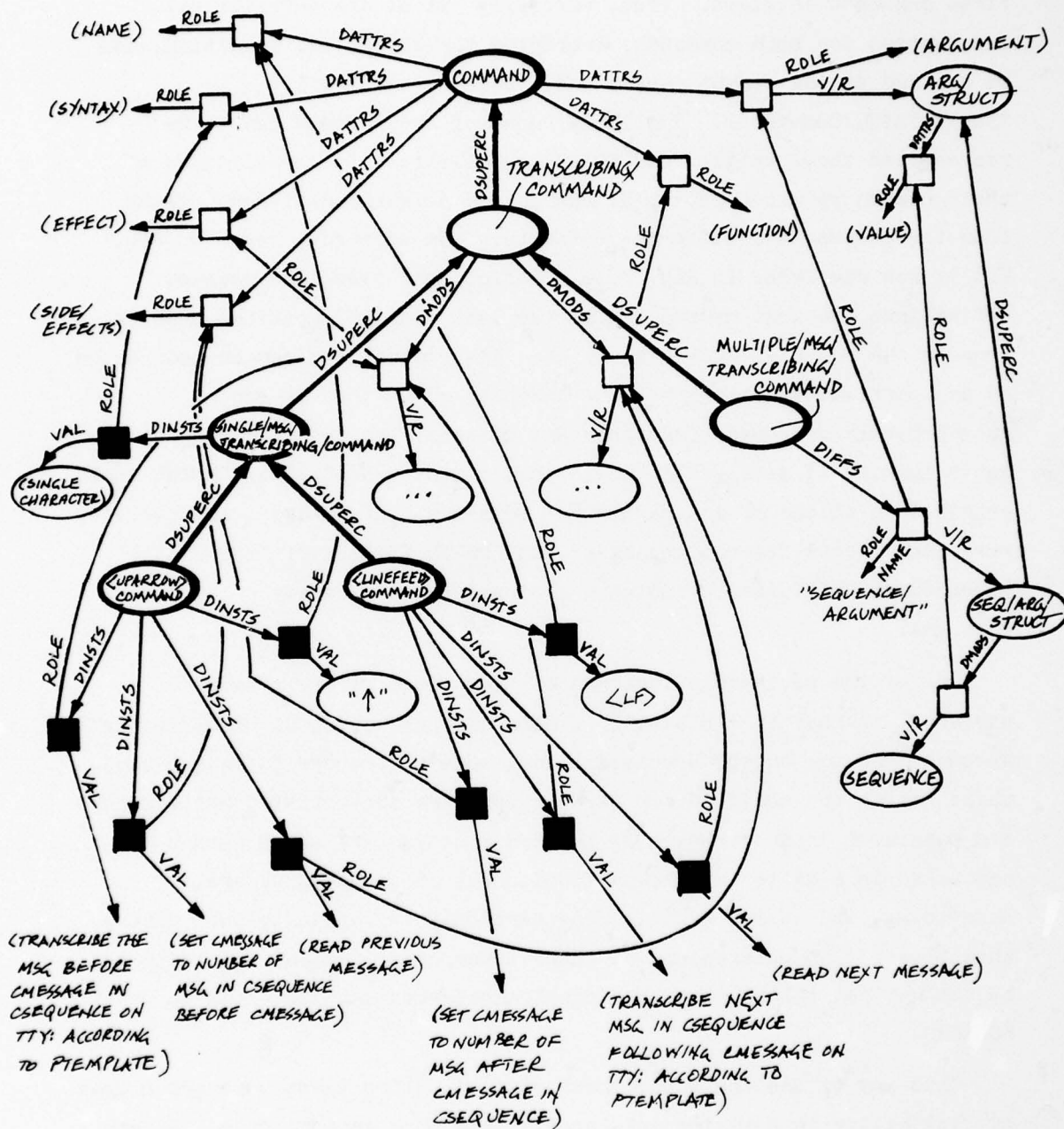


Figure 7.9. Transcribing commands, part 2.

first argument is given. Thus, virtually all of the definitional information for both commands, including the structural condition, can be amassed at the common parent node (which I will call "SUMMARIZING/COMMAND"). Similarly, most of the information to be represented about PRINT, TRANSCRIBE, and LIST can be consolidated at their common parent node. LIST has only a slightly different effect from the others, and different defaults. The hierarchy begun in Fig. 7.8 is now completed in Fig. 7.10. Notice that since the command definitions are kept merged "until the last minute", routines used to process the net will know exactly when discriminations can be made based on any particular criterion. For example, given that we can appropriately represent functions and effects, the question "How do I get a summary of messages?" should lead to the SUMMARIZING/COMMAND node, rather than either of its particular commands. If however, the question were asked about "recent messages", the DEFAULT/VALUE of SUMMARIZE's SEQUENCE/ARGUMENT should indicate a single particular way of achieving that goal.

One of the noticeable features of this piece of the command hierarchy is that it exhibits no simple uniform notion of "inheritance". At each step of concept specialization, some dattr's are differentiated while others are modified, or perhaps some are instantiated while others are passed on intact. Which dattr's are modified and which remain untouched is a matter of the particular set of commands we are describing, and in the end, is dependent only on the individual dattr's themselves. The constraints on inheritance of properties are indicated by the dattr's, rather than by a single inter-concept link such as DSUPERC.

This way of dealing with inheritance of dattr's gives us a great deal of flexibility in deriving subconcepts from more general ones. We are not forced to pass all attributes uniformly, but instead we can choose the operation that is appropriate for each dattr. This means that the DSUPERC link no longer conveys information about the modification status

Section 7.2

A hierarchy for Hermes commands

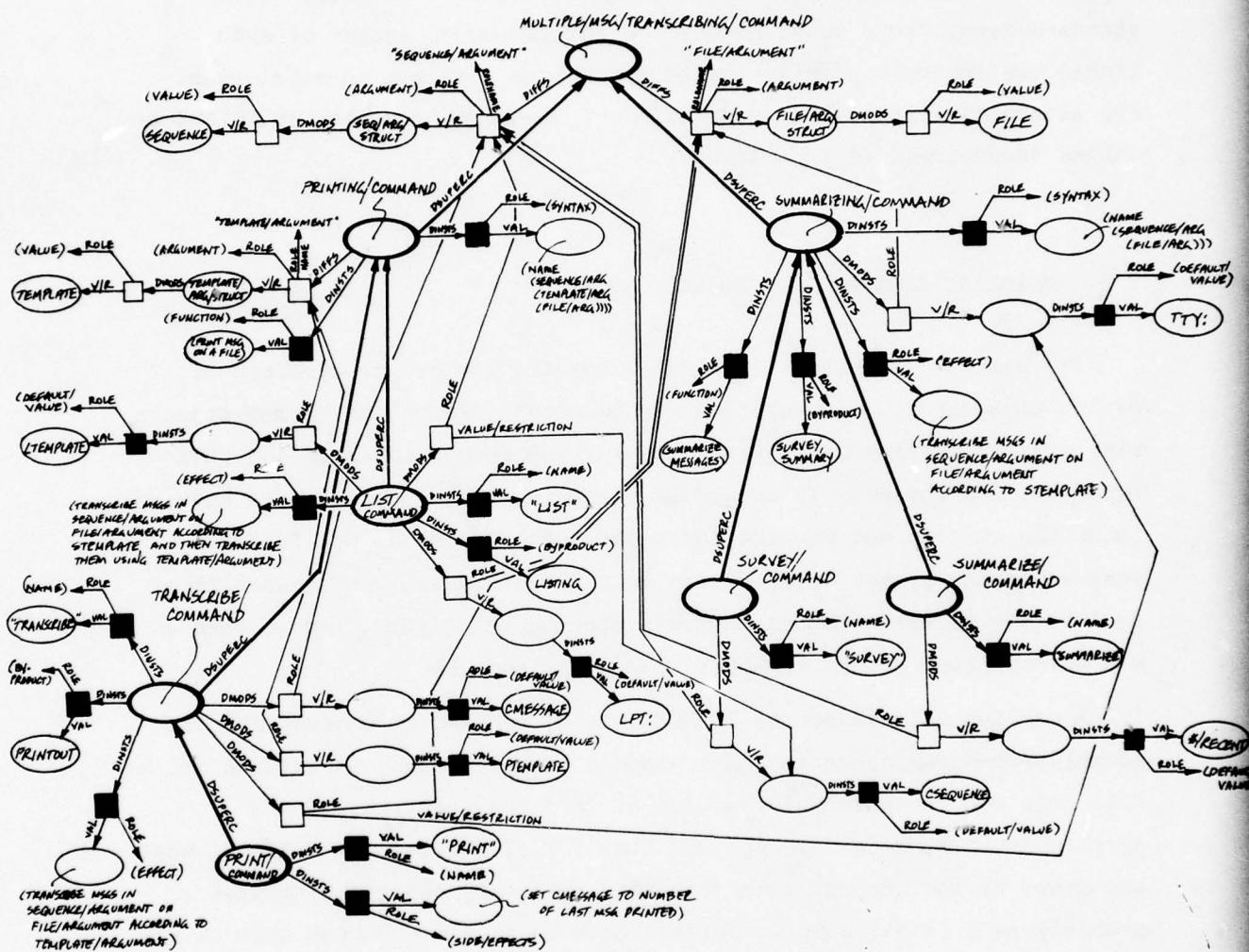


Figure 7.10. Transcribing commands, part 3.

of attributes (that information is explicitly indicated by DMODS, DINSTS, and DIFFS links). If the inter-concept link were really expected to carry the import of inheritance that it normally is in standard nets, there would have to be an open-ended number of such links, one for each combination of operations possible on attributes. The alternative approach, exemplified by SI-Net representation, might be called "decentralized inheritance".

7.3. Bringing commands and objects together

The last two sections illustrated how the structural paradigm of dattrrs-plus-structural condition may be used to represent Hermes objects with complex internal structures, as well as Hermes commands and their important attributes. It should be clear, however, that despite knowing about the objects and knowing about the commands, we do not fully comprehend the system until we can detail the connections between them. As I have mentioned, Hermes commands operate on objects, and we need a way to represent accurately what the operations are.

I earlier introduced the EFFECT role of COMMAND to account for the actual procedural operation of a command. What I meant to capture in this role was an examinable version of the code that runs when a particular command is invoked. In Fig. 7.7, I skirted the issue of what was meant by the concept of a program's effect by abbreviating what probably is a fairly complex definition with an unstructured node (i.e., EFFECT). While I shall not here attempt a complete formal description of the notion of a procedure in SI-Net notation, I will try to see how we might usefully represent runnable code in the same notation that we used for commands and objects.

A key observation to be made about SI-Net notation is the concept's resemblance to the procedure definition in many programming languages.

Each role description node can be thought of as a formal parameter, and the structural condition as the body of the procedure. Thus, an individuator reflects an invocation of the procedure by specifying bindings between values (actual parameters) and the formal parameters, allowing the body to be run on a particular set of arguments. If the procedure is a function, then its "RESULT" dattr represents the value returned by the function. In all cases, the structural condition describes how the formal parameters are to be manipulated, and the relationship between the parameters and the value to be returned, if any. Interrelationships in the notation are built out of other concepts, so that the relationships between the parameters can be examined. A concept with a structural condition could easily represent a procedure whose body is some combination of calls on other procedures*.

To represent complex effects, then, all we need do is build their structural conditions out of more basic effect "pieces", which themselves can be built out of effect pieces, down to the level of whatever basic effects we decide to be primitive. This is simply the same representational notion we have been advocating all along, here applied to the domain of program pieces. What this application implies is that we will have to have another chunk of our knowledge base, in addition to those describing objects and commands, which deals in a similar fashion with what commands do.

Given the marked similarity between SI-Net nodes and procedures, it should be clear how an effect hierarchy could be constructed. But let us look at a brief example to assure this clarity. Say that we have a primitive Hermes effect called "TRANSFORM/MSG/THRU/TEMPLATE", which takes a message and a template and yields a piece of text (the

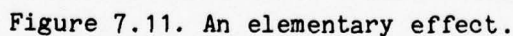
* There may be at some point a need to introduce a primitive type of structural condition, which would be a coded, non-examinable function. This is discussed briefly in Chapter 9.

transformation of the message as specified by the template). Such an operation could, of course, be described in terms of more basic operations like STRING/COMPARE and STRING/OUTPUT, but let us say that for this application, it is not useful to describe message processing below the level of TRANSFORM/... . The concept for TRANSFORM/... might then have two dattrs, one for the message argument and one for the template. The structural condition for this node would perhaps be a procedure rather than a complex of other concept nodes.

Now, all of the transcribing operations in Hermes are based on a single routine that takes a single message, a template, and an output file, and outputs the text produced by a TRANSFORM/MSG/THRU/TEMPLATE operation onto the specified file. Thus, the important effect TRANSCRIBE/MSG/ON/FILE would need to reference TRANSFORM/... in its structural condition, as in Fig. 7.11. In a similar fashion, each of the effects of the particular commands could then be built from pieces like TRANSCRIBE/..., so that they ultimately describe what those commands do in terms of lower and lower level operations.

From Fig. 7.11, we can see that the binding of the formal parameters of a higher-level routine, like TRANSCRIBE/MSG/ON/FILE, to the proper argument slots in a called subroutine, like TRANSFORM/MSG/THRU/TEMPLATE, is a simple matter of connecting the right role description nodes of the enclosing concept with the appropriate coreferential role nodes of the paraindividuals in the structural condition. While this is an easily glossed-over matter of binding, we can see that its impact is more significant than first appears when we consider more carefully the meanings of those nodes and links. This impact becomes apparent when we step back to our starting place for effects -- the notion of a COMMAND.

Recall that the EFFECT role of COMMAND was essentially to capture what the command did to its arguments. The arguments themselves were specified by role description nodes, one for each argument. For example, the PRINT/COMMAND has (ultimately) three associated ARGUMENT roles inherited through its DSUPERC chain (see Fig. 7.10). Each of the



three role nodes constitutes a description of all the potentially legal fillers for that role (i.e., the concept pointed to by the VALUE/RESTRICTION link implicitly defines a possibly infinite class of entities, whose members may or may not be known at any given time, and all legal fillers of the role must ultimately come from that class). Since the concept nodes in our network are descriptions that are applied only to individual entities (i.e., a concept, while implicitly defining

an extensional class, is applicable as a description to only one individual at a time), the role nodes are in the same way applicable only to individual fillers. They implicitly define the whole set by circumscribing the characteristics of any single member.

If one of these nodes constitutes the description of an individual potential filler for the role -- some indefinite, singular entity that we cannot specify in advance -- then what does it mean to point to, to reference such a node? Consider again the simple binding we encountered in Fig. 7.11. The role nodes for the message and template of TRANSCRIBE/MSG/ON/FILE describe singular but unknown entities which will fill those slots in any instance of that effect. From the paraindivuator of TRANSFORM/MSG/THRU/TEMPLATE in the structural condition are COREFVAL links to each of these role nodes, and as stated earlier, this binding works like that of a reference to a formal parameter within the body of a procedure. Neither the particular template passed to the TRANSCRIBE/... routine nor the particular one passed into the TRANSFORM/... routine need be specified in advance for the definition to make sense. In the case of procedure definition, then, we are taking advantage of the formal parameter's ability to serve as a placeholder for any entity passed in as the argument -- that is, as an abbreviation for all potential argument fillers (but referencing it as an individual).

The COREFVAL link between a role node of a paraindividual in the structural condition and a role node of the defining concept is thus a statement of context-dependent intensional equivalence. Role nodes are descriptions of individual entities that can exist independent of any entities which actually fit those descriptions, and correspond to real entities when the concept is applied to some particular world. The tie is one of intensional equivalence because it states that in any world to which the concept is applied, the 2ND/ARGUMENT of a TRANSCRIBE/MSG/ON/FILE action is always identical to the TEMPLATE/ARG of the corresponding TRANSFORM/MSG/THRU/TEMPLATE, by definition. The

dependence is context-dependent because it is only true in the context of the particular TRANSFORM/... invocation. This is what is meant by a "parameterized individual" -- a context-dependent description.

The original goal in this section was to illustrate how to interconnect the Hermes command effects and the objects on which they operate. We first treated the operations as themselves structured concepts, and then examined intensional references to role fillers. Since the EFFECT of a Hermes COMMAND is one of its dattr, and so are each of the ARGUMENTs to the command, it is the job of the structural condition of each type of command to bind together the operation of the command and its operands.

Since the structure that fills the EFFECT role of a COMMAND is set up to take as its own dattr the Hermes objects that are to be affected, the structural condition has merely to set up a group of correspondences between those dattr and the particular objects on which the command operates. For example, Fig. 7.12 illustrates how we might first attempt to hook up the EFFECT of a SUMMARIZING/COMMAND (which is only schematically indicated here) to the SEQUENCE and FILE taken as arguments to the SUMMARIZING/COMMAND. The EFFECT dattr is represented by node E, the SEQUENCE dattr by node S, and the FILE dattr by node F. Notice that an object not associated in any other way with SUMMARIZING/COMMAND (STEMPLATE -- in the lowest part of the figure) can still be bound into the effect description. In this manner, the structural condition serves to capture inter-role relationships by explicitly stating bindings between different subpieces of intensional structure. This is the purpose of the (EQUIV) nodes in the structure of Fig. 7.12.

In Fig. 7.12, there are problems with some of the substructure references. The COREFVAL links from the EQUIV paraindividuators into the EFFECT of the SUMMARIZING/COMMAND point to the role description nodes of TRANSCRIBE/SEQUENCE/THRU/..., and not to the EFFECT role node

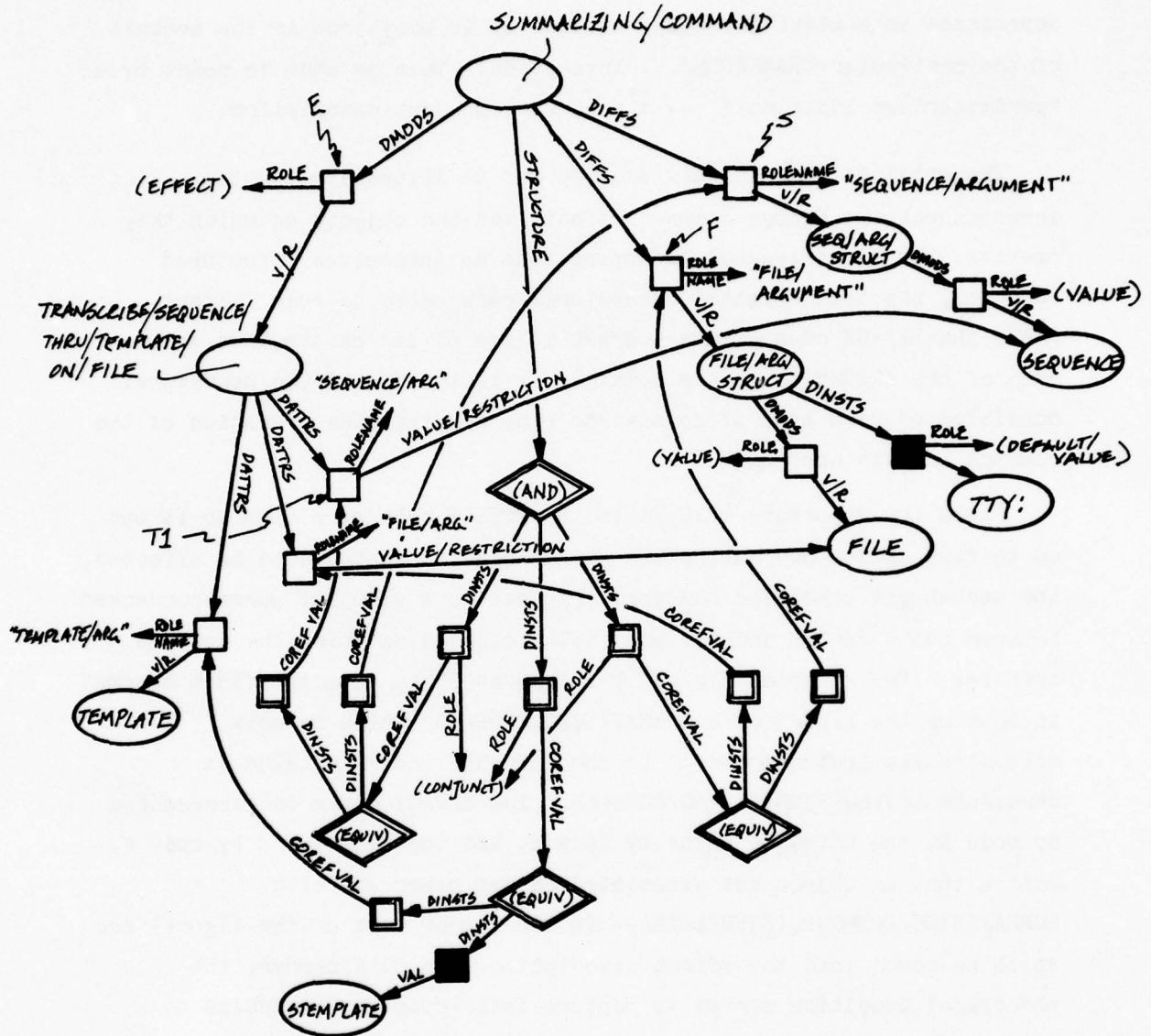


Figure 7.12. A first attempt at synthesis.

of SUMMARIZING/COMMAND. On the other hand, the pointers to the ARGUMENTs do point to the role nodes of SUMMARIZING/COMMAND, but recall that, in the definition of COMMAND (Fig. 7.7), an argument was a structured entity with two parts -- its VALUE and its DEFAULT/VALUE. The command operates ultimately on the VALUE of the argument, not the

"ARG/STRUCT" itself. That is, the SURVEY/COMMAND prints a survey of certain messages, not one of certain arguments. Thus, we really might want to have two pointers from the structural condition of SUMMARIZING/COMMAND not to the SEQUENCE/ARGUMENT and FILE/ARGUMENT nodes (S and F, respectively), but to the VALUE role nodes of each of their VALUE/RESTRICTION concepts (thereby stating that it is the VALUE of the sequence argument that is ultimately affected during command execution, etc.). However, a single pointer to one of the VALUE role nodes would run into the same problem as the ones into the EFFECT do -- they specify the right pieces of substructure, but do not tie them down to their reference from this concept.

What is needed here is a pointer to a substructure of a concept in context. It is surely the VALUE of some SEQUENCE/ARG/STRUCT that is operated on, but only the one referenced in the context of SUMMARIZING/COMMAND. The node, SEQUENCE/ARG/STRUCT, is a concept node like any other, and can therefore be pointed to by many other nodes in the network. To point directly to SEQUENCE/ARG/STRUCT from the structural condition of SUMMARIZING/COMMAND would entail making a statement about all SEQUENCE/ARG/STRUCTs, in all contexts. It is not the value of the sequence argument in any other case which is of concern to SUMMARIZING/COMMAND. The node for SUMMARIZING/COMMAND only defines what happens to its arguments (which is why each concept has its own unique set of dattr's, even though they refer with VALUE/RESTRICTION relations to generally referenced concepts).

To be more precise, the COREFVAL pointer from node T1 in Fig. 7.12 to the concept node SEQUENCE states that a sequence is used in the invocation of TRANSCRIBE/SEQUENCE... caused by the summarizing command. It does not constrain the sequence, however, to be the one passed in as the value of the argument to the command, since it points to the general concept of a sequence.

What is needed here is a two-part access, to pin down the particular sequence argument required and then to indicate the appropriate part of

the "argument" (i.e., its VALUE). Once again, this is the purpose of the "structural reference nodes" introduced in Chapters 4 and 5. The FOCUS link picks out the particular dattr of the defining concept on which we want to concentrate; the SUBFOCUS link picks out the subpart of the filler of that dattr. So, for example, the binding of the sequence input to a SUMMARIZING/COMMAND and the one required in TRANSCRIBE/SEQUENCE... in its structural condition would be resolved as in Fig. 7.13. The FOCUS link determines the context and the SUBFOCUS

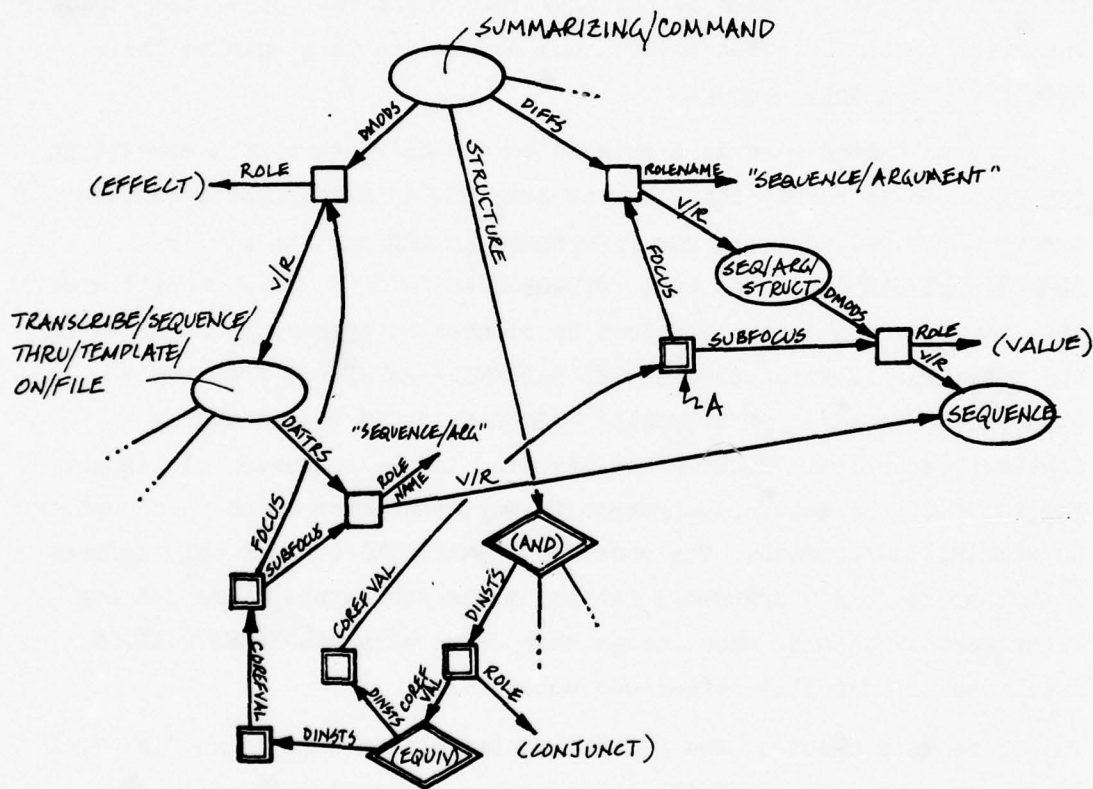


Figure 7.13. Substructure reference.

link determines the substructure. Notice that a structural reference node indicates a path through the structure -- node A says "the VALUE of the SEQUENCE/ARGUMENT of this concept". Such paths are not necessarily a single level, since, for example, the REPLY command uses the CONTENTS of the SUBJECT: of the VALUE of its MESSAGE/ARGUMENT to construct the

subject of its result. This path reference would require several structural reference nodes linked together (see Fig. 7.14).

One final observation we might make about the interaction between commands and objects is that objects might be considered principally as participants in actions, rather than solely as entities into themselves. For example, we might want to define a MESSAGE as a participant in some kind of Hermes communication act. Such a definition would look quite different from the static, object-oriented ones that we have described. A MESSAGE in this view would have as its dattrrs things like AUTHOR, RECIPIENTS, etc., and its structural condition would juxtapose concepts like COMPOSE and SEND (Fig. 5.2 illustrates a possible account for this structure). Such a representation resembles those of many action-oriented English nominals, as discussed earlier. SI-Nets allow the simultaneous definition of an entity as a static structured object and as a participant in certain relationships and actions.

7.4. Keeping track of the Hermes environment; Individuation

While it may be possible to describe completely the Hermes program in a way that would allow an intelligent agent to talk about it, it would be less than optimal if the agent could not use that knowledge to discuss the user's current environment. One of the most important uses of a knowledge base such as the one we have been discussing is its facilitation of the interpretation of the objects in the world around it in terms of concepts embedded in that base. A Hermes user will invariably want to talk about his messages, his templates, and the particular command invocations he wants to give.

To this end, the Structured Inheritance Net offers the notion of individuation -- the derivation from a general concept of the description of a particular individual. As discussed at length in Chapters 4 and 5, we must be extremely careful about the notions of

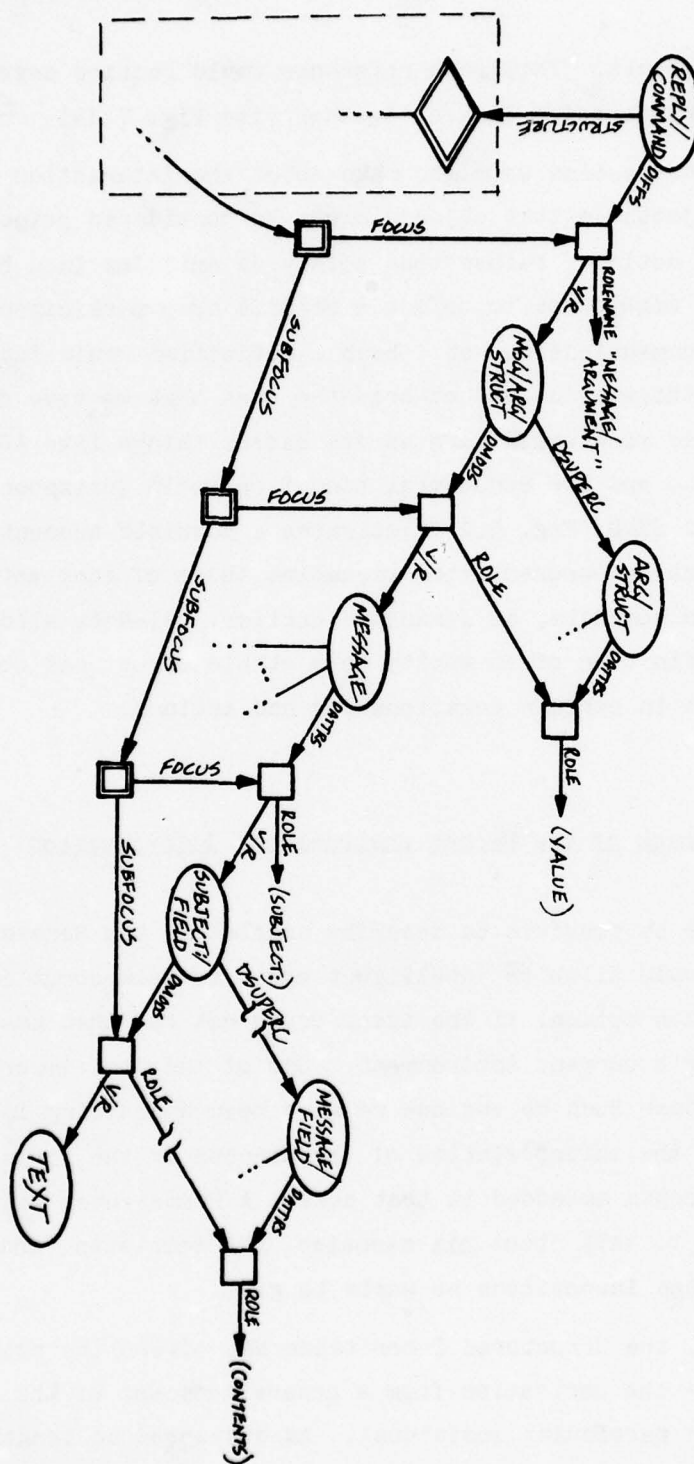


Figure 7.14. A deeper substructure reference.

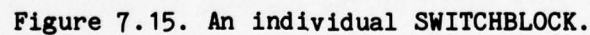
"individuator" and "instance", since individuators are intensional entities rather than the extensional or hybrid ones for which they are most often mistaken (see Section 4.3.3). In this section I look briefly at how the SI-Net representation makes a discussion of the current Hermes context possible through its very general mechanism for individuation.

Most case-like semantic net notations (and other related formalisms -- see Chapter 8) offer the following paradigm for deriving descriptions of instances: a "concept" specifies in some way a set of slots or cases, each of which defines a piece of a structured object. For a particular object to be considered as an instance of the concept, it must have "parts" which fill in each of the slots in the appropriate manner. In SI-Net representation as well, a concept node embodies the constraints on a single entity, and we would expect an individuator representing an instance to manifest a set of case fillers that map one-to-one onto the cases of the concept*.

So, for example, to represent a user's particular set of switches, we need merely account for all of the roles associated with a SWITCHBLOCK (see Fig. 7.3) with a set of role/filler pairs, specified by DINSTS links. For each dattr of the concept of which the description of the switchblock is an individuator, we provide a DINSTS link to a role instance node, which in turn details how the role is filled. Thus, a switchblock would be individuated as in Fig. 7.15.

By the same token, any other particular object of concern to the

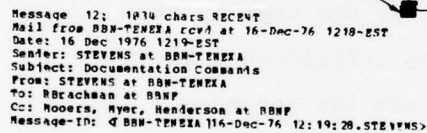
* I have introduced some new features here, such as specifying the important attributes of an entity, not just its parts; accounting for the structure of a structured object explicitly; a role differentiation capability (so individuator role fillers map many-to-one); and particularizing role fillers at the concept level (with DINSTS); but the general idea is still the same -- in this notation as well as others, individuators are filled versions of schema defined by concepts.



user might be represented by an individuator of the relevant concept type. Figure 7.16 illustrates a message and a template. The important thing to notice here is the uniform way to derive individuators from concepts, dependent only on the small, well-defined set of primitive link types. All concepts defined with DATTRS, DMODS, DIFFS, and DINSTS links have a well-known algorithmic way to be individuated (i.e., dattrs and differentiated dattrs are filled by individuators of the concepts pointed to by their VALUE/RESTRICTION links, modified dattrs are filled by individuators of the VALUE/RESTRICTION concept derived from their own links and those of their source roles, and instantiated dattrs are

Section 7.4

Individuation



In addition to the changes suggested in Charlotte's memo of 16-Dec-76, I think that for consistency's sake we should default document to depth of 2.

As far as the suggestion about using DESCRIBE to get at things like User's-own-objects, I think that we should think about that some, since that broadens the meaning of DESCRIBE significantly beyond a mere documentation command.

I also just thought of one problem with the DESCRIBE Index scheme. How does one use an index that does not have page numbers? I'm not sure we want to get into making a Document command sophisticated enough to remember where all of the topics are and then generating the page-referenced index afterwards.

---A1

```
>show sample
(1) "SS100-V0.1" 804000
(2) "----" CH1-Coup: 8040-0100: 8040:
5
```

-233-

filled already by the particular values they specify).

The explicit role filler to role definition links provide the connections necessary to discuss the meaning of each of the parts of an object. It is always clear which entity fills the TEXT:/FIELD role, for example, thereby allowing access to its relations to other entities and the message as a whole. The connection to the TEXT:/FIELD role description node of MESSAGE also yields a path to the definition of a FIELD in general; this allows several alternative descriptions of the same piece of text, and would allow an intelligent agent to answer questions about the text of a message worded in different ways (i.e., using all of the intensional connections available from nodes like TEXT, MESSAGE, and FIELD).

In addition to tying descriptions of particular objects to the defining concepts, an intelligent program could follow and talk about command invocations in a similar way. For example, if the user types "SURVEY RECENT", the SURVEY/COMMAND concept would be individuated as in Fig. 7.17. The VALUE of the SEQUENCE/ARGUMENT becomes the sequence RECENT, the FILE/ARGUMENT is defaulted to TTY:, and the template used for the transcription is the one always used, STEMPLATE (not shown in the figure -- it is specified in Fig. 7.12). Given these roles filled this way, it is possible to tell what will happen (through the EFFECT -- remember the bindings are made through the structural condition of the general concept, SURVEY/COMMAND), and to discuss to some extent the SIDE/EFFECTS, BYPRODUCTS and potential FUNCTIONS for this particular invocation.

It is a bit of an oversimplification to say that when the user types a Hermes command, the appropriate command concept is simply individuated. Two different aspects of the meaning of "command" come into play here: 1) the command as something that the user types, i.e., a thing with SYNTAX, and 2) the underlying Hermes routines that are invoked (i.e., an EFFECT) when the typed syntax is translated into a

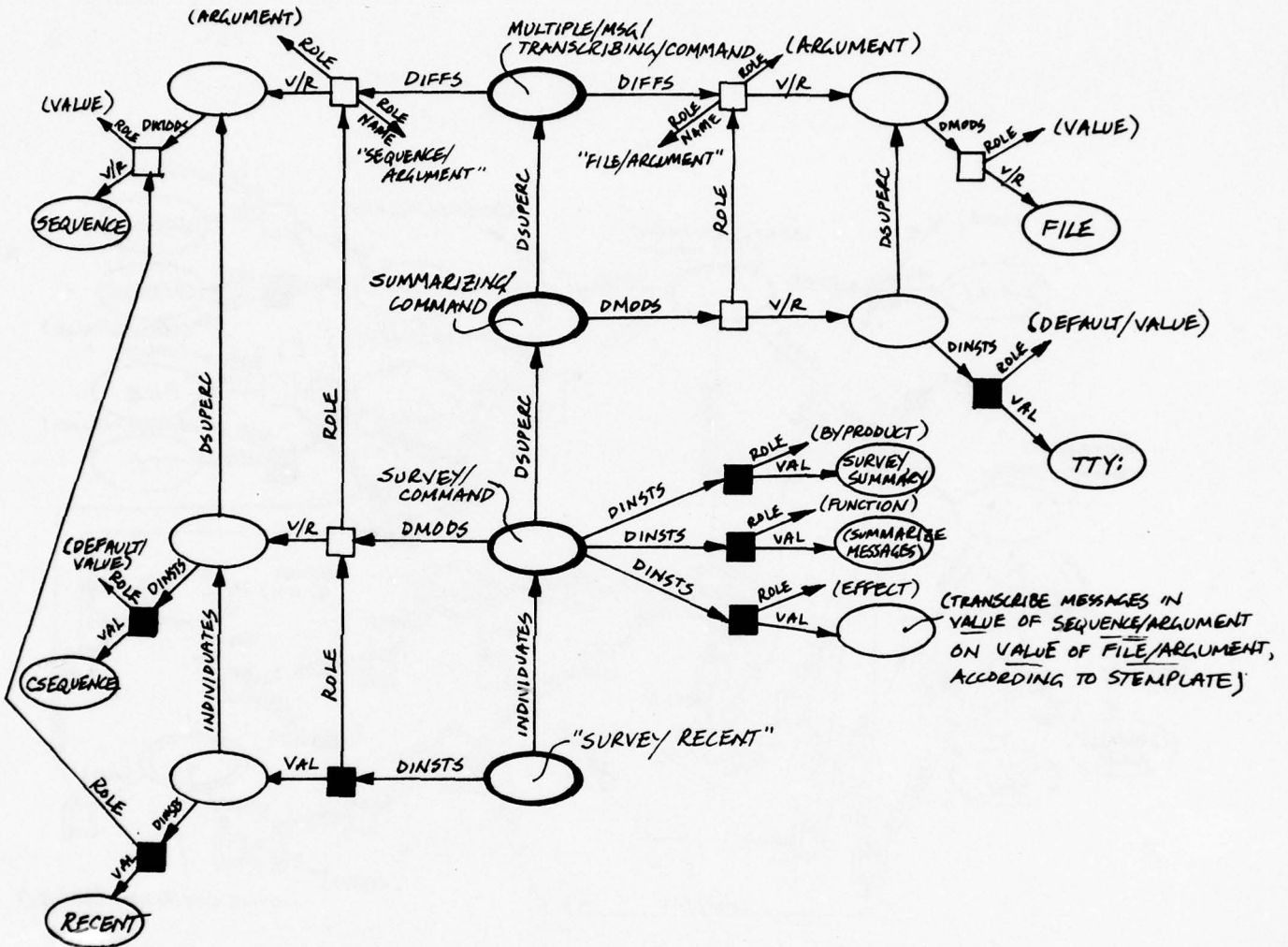


Figure 7.17. SURVEY RECENT.

command name lexeme and ARGUMENTs. Therefore, to reflect accurately the transformation that must occur to individuate the execution aspect of a command, the structural condition for COMMAND must account for the relationship between the typed SYNTAX of the command and the underlying objects corresponding to the arguments typed. Figure 7.18 suggests how this transformation might look; we leave to future research the detailed investigation of such connections.

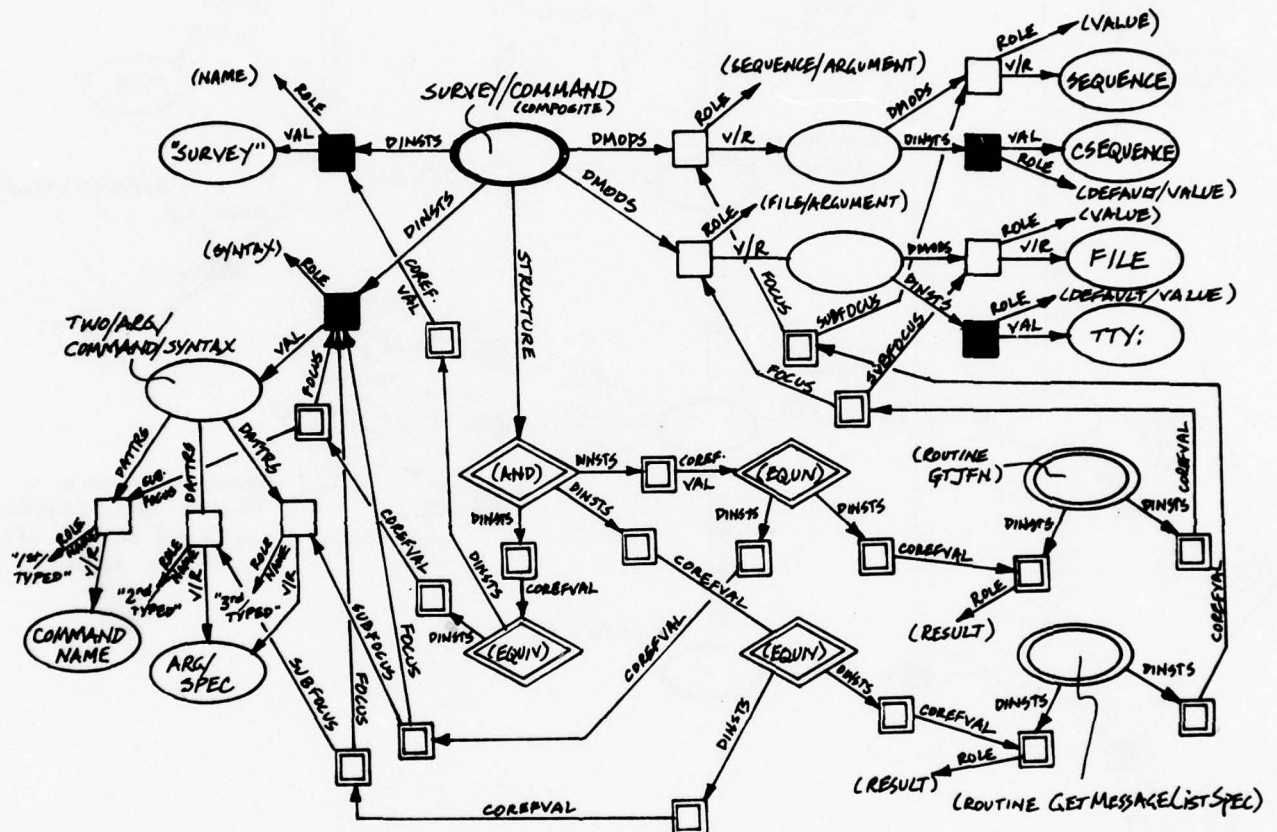


Figure 7.18. Transformation between command aspects.

Chapter 8. An Analysis of Current Representation Methodologies

In the last several years, it has become increasingly apparent that an Artificial Intelligence program cannot perform intelligently as a tabula rasa -- rather, a great deal of knowledge must be supplied to the system before it can begin its appointed task. With this increased appreciation of "knowledge-based programming" came the realization that the representation structure in which the knowledge was encoded itself had a major impact on the ultimate success of the program. As a result, a small number of projects have recently been initiated to study representation languages as things in themselves. Among these we might include the study of Frames, initially motivated by Minsky's 1975 paper and continued by Ira Goldstein and Bruce Roberts as FRL at M.I.T.; the KRL project, undertaken by Dan Bobrow, Terry Winograd, and the Understander Group at Xerox-PARC; the development of the MDS system by Srinivasan at Rutgers; research on the English-like OWL language by Bill Martin and his Automatic Programming Group at M.I.T.; investigation into state-based (Cercione and Schubert) and partitioned (Hendrix) semantic nets; and the older MERLIN paper of Moore and Newell* (and of course, the work in this report, in addition to Bill Woods' "What's in a link" paper).

The striking thing about this group of projects is the convergence of intuitions about the structure and use of "chunks of knowledge" which has been developing over the last year or two. This Zeitgeist includes thoughts on structured conceptual entities, which have closely related

* A detailed criticism of this effort is included in an earlier paper of mine [1975].

pieces of information designated by "slots", ideas about the use of such patterns as prototypes, which serve as the source of default knowledge about groups of entities, and associated notions of inheritance, pattern-matching, searching, individuation, and procedural attachment. In this chapter, I would like to investigate in detail this emerging picture of knowledge structure, and see how some of the other projects stack up in terms of the methodology and representation issues discussed in Chapters 3, 4, and 5.

Here I shall concentrate on three of the major representation methodologies, investigating in some depth KRL [Bobrow & Winograd 1977, Smith 1977], MDS [Irwin & Srinivasan 1975, Srinivasan 1976], and FRL [Goldstein & Roberts 1977, Roberts & Goldstein 1977]. While each of these really comprises an entire knowledge representation system (of which the representation language is an integral part), I will focus on the representational primitives offered by the KRL, MDS, and FRL formalisms, deliberately ignoring the elaborate environments of which they are part*. The hope is to understand the common themes of representation, to distill out the important ideas generated by these efforts, and to see how they really differ from one another. To this end, our own role-oriented SI-Net formalism will serve as a language in which to discuss the details of the other systems. I will walk through many of the important points raised in this report, and see how each of the languages handles the tasks of representation set out in Chapters 4, 5, 6, and 7. We will see that to a large extent, the representation scheme developed here can be thought of as a distillation of many of the important points buried in these systems (although it seems to account for several points not evident in any of the other languages). Our

* For example, "MDS", which stands for "Meta-Description System", is a general system for building problem-solving systems and includes a general theorem-prover and a general GPS-like problem solver. As I have mentioned, I am here interested in only the representation language used in the system.

insistence on explicit representation of underlying primitives and conscious imposition of an epistemology will help us to understand better the other knowledge representations of the day.

I will begin the discussion by introducing the surface forms of the representation languages of KRL (which is in the process of revision, and I include treatment of both KRL-0 and KRL-1), MDS, and FRL (FRL-0). I will then proceed to a more detailed discussion of the representational primitives underlying these languages, and, eventually, I will look beyond the structures themselves to the semantics of the representations. This last level of analysis will expose the strengths and deficiencies of these representations with respect to the set of issues raised in Chapters 4 and 5. Finally, I shall seek to understand the differences in worldview imposed on users by each of the frameworks, and determine how helpful the epistemologies imposed by the authors of the systems are for knowledge representation in general.

8.1. Language forms

The constructs of KRL, MDS, and FRL differ from those of semantic net-type formalisms in one glaringly obvious respect -- one usually expresses structures in networks in graphic form, while those expressed in the other formalisms are in lexical form. Ultimately, these two surface forms are equivalent, and it is as easy to express, for example, an MDS template as a network node with links as it is to express a semantic net in relational triples. However, the styles of use of these two types of notation vary, and the way that one "sees" his knowledge base can be influenced considerably by the modality of the language expressions. Therefore, it is worth considering, at least briefly, the advantages and disadvantages of both types of notation.

8.1.1. Basic language constructs

KRL and FRL use LISP-like notations, with parenthesization playing a large part in surface forms. An obvious problem with this type of notation, resolved to some extent by "pretty-printers", is the lack of perspicuity of deep, highly-structured items. Beyond a few levels it becomes very difficult to sort out lists of lists of lists, and with the current move to highly structured objects, a look at a deeply-nested set of concepts yields only headaches. Very careful typing is necessary also, to insure against misplaced structures*.

One way to combat this problem, used often in the structures actually implemented in these languages, is to "subroutinize" the structures. Rather than include, in line, a complex definition of a frame as a subpart of another frame, a separate structure can be created, named, and pointed to from the original. The same goes for procedures used within a frame; these pointers substantially alleviate the problem of notation readability. However, this makes names very important**, and makes structures clear only by continuous cross-referencing.

This is particularly a problem with the more constrained MDS notation, where each relational statement must name a destination. For example, consider the template definition of Fig. 8.1 [Irwin & Srinivasan 1975, p. 74]. The template being defined is STATEDESN ("state description"), the description of a state in the diagnosis of a disease. On each line of the template description is a relational

* On the other hand, no methods yet exist for "typing" graphic network structures. What we need is a "habitable surface language" for communicating to a computer the structures we have in mind -- a way to write down concepts in a machine-readable form. No such language yet exists for the notation presented in this report.

** See McDermott [1976] for an important caution about names in AI systems.


```
[TDN: (STATEDESN TN)
      ((startingweight !)(PROB T#) startingweightof CC7)
      ((descendents X)(STATEDESNS $L) descendantsof CC8)
      ((causes !)(CAUSEDESNS $L) causesof CC10)
      ((status C)(STATUS TI) statusof CC11)
      ((conflict C)(CONFLICT TI) conflictof)
      ((presence C)(PRESENCE TA) presenceof CC13 TR2)
      ((likelihood C>!)(IT LIKELIHOOD NIL) likelihoodof)]
```

Figure 8.1. An MDS template.

statement with four parts. The first, a parenthesized pair, is a binary relationship that the current template participates in with some other template. The first element of the pair names the relationship, the second is a "flag" for the interpreter. The second pair on the line specifies the kind of template which is the other participant in the binary relationship. The (all upper case) first element names the template and the second is a special item that determines the type of the template. In this figure, the third line of the definition states that any STATEDESN will stand in the "causes" relationship to some CAUSEDESNS ("cause descriptions"). CAUSEDESNS is itself a template, with a structure somewhat like the one in the figure*. The "\$L" means that CAUSEDESNS is a "list" template -- it is really a set of single CAUSEDESNSs. The third element on the template line is the inverse of the relationship that starts the line; all relations in MDS have inverses. Finally, the last element names a "consistency condition" (CC) to be applied to the filler of a relation in an individuator. We discuss these CC's below.

In this notation, the destination of a relation must be a single

* For now, we can think of templates in MDS as concepts in SI-Net notation, with relations like "causes" being the roles of dattr of a concept. The equivalent of a role description node -- the source of the ROLE link in our notation -- is the pair (STATEDESN causes). This is called an anchor in MDS. Thus, the template pointed to in each line -- the second structure on the line -- is the VALUE/RESTRICTION of that dattr.

template. Thus, to "see" the overall structure of a STATESDESN, with the structures of each of its parts, one must at least look up the definition of CAUSEDESNS, and then its component templates; in all likelihood, this will entail thumbing through pages and pages of template definitions (see the appendix of [Irwin & Srinivasan 1975], for example).

In fact, one good way to understand the structure of data bases expressed in KRL, MDS, or FRL is to draw pictures of the interconnections among structure definitions. The expressive power of the graphic notation, at least as far as structure is concerned, should be obvious from the multitude of figures in the last two chapters*. Notice that things like node type are easily distinguished by shape in the graphic notation, while in lexical forms they are merely more words in special places. For example, consider the KRL-0 units in Fig. 8.2 [Bobrow & Winograd 1977]. The unit Person and the unit G0043 are of two

```
[Person UNIT Basic
  <SELF>
  <firstName (a String)>
  <lastName (a String)>
  <age (an Integer)>
  ... ]

[G0043 UNIT Individual
  <SELF {(a Person with
    firstName = "Juan"
    lastName = {(a ForeignName)
      (a String with firstCharacter = "M")}}
    age = (which IsGreaterThan 21))
  (a Traveller with
    preferredAirport = SJO
    age = Adult)
  ... }>]
```

Figure 8.2. Two KRL-0 units.

* Although, admittedly, the more links one adds to the net, the more confusing the picture. Layout becomes the critical issue.

different types. These types are specified by "Basic" and "Individual" indicators. Notice the difference in impact between these two markers and the differently shaped nodes in, say, Fig. 7.6.

The progression to KRL-1 has seen the elimination of unit types and the introduction of less LISP-like unit structures. In the newer system, units would appear as in Fig. 8.3 [Smith 1977]. In these (and

```
#Person
  self: an Animal
        an IntelligentBeing
  age:  a Number
  sex:  Or(Male,Female)
  name: a String
...

#Aaron
  self: a Person with age = 26
                        sex = Male
...
```

Figure 8.3. KRL-1 units.

the above) units, parts of a unit are designated by its slots -- in the earlier notation these were indicated by angle brackets (" $<$ ", " $>$ "), and in the newer notation are simply listed with the unit name, each slot name being followed by a colon (" $:$ "). Associated with each slot is a description, comprising a set of descriptors. These descriptions describe the potential and actual fillers of the slots. Thus we see that a Person has a firstName which is a String, and an age which is an Integer (Fig. 8.2). The special self slot is used to describe the unit as a whole -- Aaron, as a holistic entity, is a Person whose age is 26 and whose sex is Male (Fig. 8.3). The type of descriptor that appears in Aaron's self slot is a perspective. The idea behind it is to view Aaron as a Person, and to note the differences between Aaron and the stereotypical person. Person here is the prototype in this description by comparison, and the equivalences following the keyword "with" are considered to be further specifications of that prototype.

It is hard not to notice the strong English-like character of the KRL language. Keywords like "a", "the", "whichIs", "from", and "thatIs" are common in KRL-1. This is a tremendous aid to the readability of the notation, and thankfully eliminates the parenthesis headaches of KRL-0, MDS, and FRL. However, we should caution against the reliance on a language that is close, but not identical to a more familiar language, especially a natural language, with its multitude of idiolects and idiosyncrasies. The user must be aware at all times of the precise formal definitions of each of the KRL keywords, and constantly remind himself that he is not speaking English, but KRL. It is very easy to fall back on the more familiar language without even realizing it. How can we remember the formal distinction between "whichIs" (introduces a functional -- a predicate) and "thatIs" (an abbreviation for "with self ="), when, as English phrases, they are so similar in meaning?

As in KRL, the frames of FRL have slots. A frame is really only a named list of such slots, with the slots themselves having, possibly, some further list structure. In Fig. 8.4*, we see a frame for the

```
(fassert LUNCH
  (ako ($value (eating)))
  (schedule ($default (shareable (with: communication))))
  (time ($prefer ((overlap? (daytime :value)
                             (interval (noon) (pm 1)))))
    ((when duration) ($prefer ((between? :v (hour .5)(hour 1.5)))
                              ($default ( @ (hour 1)))))
  (place ($prefer ((ako? :value 'restaurant)))))
```

Figure 8.4. A frame for LUNCH.

concept of "LUNCH". The first item in each slot list is its name, the first item in each sublist of the slot is the name of a kind of property, called a key (or aspect or facet -- these are by convention

* The FRL examples in this chapter are adapted from the code for the "Nudge" system [Goldstein & Roberts 1977] -- see [Roberts & Goldstein 1977].

prefaced by "\$"), and the sublists following the keys are properties. So, for example, in the LUNCH frame, the "schedule" slot has one key, "\$default". Under this key is the single property, "shareable"; the "with: communication" suffix is a comment on the property, and is composed of a topic (by convention followed by a colon) and a message*. In FRL, the set of keys is not pre-defined, although certain ones are expected by processing routines, and have special interpretations (such as "\$value", "\$require", "\$prefer", "\$default", "\$if-needed", "\$if-added", and "\$if-removed").

This notation is very simple and uniform. It involves no special constructs like KRL does, yet is more general than the constrained list of triples of the MDS template. This simplicity is both an advantage and a disadvantage -- while the set of keys, properties, and comments is completely at the discretion of the user**, the only constraining syntax is the parenthesization. For a non-LISP user, a frame looks like parenthesis salad next to a KRL-1 unit.

8.1.2. Procedures and constraints

Another noticeable feature (both advantageous and disadvantageous) of the simple FRL syntax is the direct incorporation of procedural information. In Fig. 8.4, the preference for the "place" slot is "anything which is a restaurant". In SI-Net notation (cf. the VALUE/RESTRICTION) and in KRL, the explicit "ako?" predicate is not required, since the pointing of slots to other nodes is interpreted automatically by the system as class restrictions on potential fillers

* The key-plus-property pairs here are like the links we have emerging from role-description nodes in SI-Net notation.

** In all of the systems that are being considered here, the slots are user-defined, except for "ako" and "self".

(in the SI-Net case, the interpretation is constrained by link type; in KRL, by place in the structure and keyword). Since particular keys in FRL are not part of the defining syntax, the predicate call must be explicit, to distinguish it from a direct pointer such as the "\$value" property of the "ako" slot.

In the above example, "ako?" is a LISP function -- the FRL language does not itself provide a syntax for procedure definition. This allows the user the full power of LISP in his frame system, with no constraints on what values are accessible by procedures. But it also forces the user into the LISP domain extremely often, with a resultant proliferation of supporting LISP code, not provided by FRL itself. For example, cross-slot agreement routines are not part of the FRL language, and each dependency must be handled by new routines written by the user. In Fig. 8.5 we see a frame with an "if-added" method (in this frame,

```
(fassert ONE-WAY-COMMUNICATION
  (ako ($value (communication)))
  .
  .
  .
  (participant ($default ( @ (union !!to !!from)))
    ($if-added ((add-from-or-to)(finherit: continue)))
    ($require ((forall?
      (or (member :value !!to)
          (member :value !!from)))
      (type: agreement)))))
```

Figure 8.5. A frame with an "if-added" method.

":value" means the current value for the current slot, thus the value that is being added to the "participant" slot, and "!" means take the value of the slot indicated). If a "participant" is to be added to an instance of ONE-WAY-COMMUNICATION, the procedure "add-from-or-to" is to be run. This is some arbitrary piece of LISP code set up by the creator of this frame for keeping slots which depend on each other consistent. Notice how procedure calls fit right into the syntax in abbreviated (name) form, just like the earlier cases of structure pointers, or in

fuller form, as in the "\$require" property, which checks agreement of the participant slot with the "from" and "to" slots of the same frame. Since these procedures are not really defined in FRL, but rather in LISP, we must understand LISP to know what this frame is all about, and again, perspicuity is at the mercy of names*.

KRL calls upon LISP also in a more or less direct way for procedural attachment. Figure 8.6 [Bobrow & Winograd 1977, p. 23] illustrates a

```
[Ownership UNIT Specialization
  <SELF (a State)
    TRIGGERS (ToEstablish
              (AND (Match \(the possession) \(a Dog))
                  (Match \(the owner)
                      \(which Owns(a DogLicense with
                        licensed = (the possession))))))>
  <owner (a Person)>
  <possession (a Thing)>]
```

Figure 8.6. A KRL-0 call on LISP code.

KRL-0 call on LISP for the establishment of a certain kind of ownership relationship. Notice the interleaving of LISP and KRL. Everything within the "ToEstablish" call is LISP code to be used by the matcher whenever Ownership is the prototype unit referenced in a perspective descriptor being matched. The backslash ("\") is an escape character indicating a KRL expression.

While the use of some KRL syntax within LISP code and the presence of a keyword ("TRIGGERS") to indicate non-KRL notation comes closer to

* In cases where one would expect a value, rather than a procedure, the at-sign "@" is used to invoke EVAL. Note that there is nothing in the syntax which distinguishes these cases -- their differentiation is implicit in the routines that process the framework. If the definitions of the keys were fixed by the language, then such an implicit discrimination would be acceptable. But in FRL, the user can create his own keys with their own interpreting routines, and thus one cannot tell from knowledge of FRL alone whether the properties are procedures to be evaluated or particular values.

procedural definition within the host language, MDS goes one step further and explicitly specifies a sublanguage for its procedural constraints on slots, called "consistency conditions" (CC's). While this language is not the same as the template definition language, and can therefore not be considered "known" to the system in the same way as the knowledge base, it is at least well-defined within the MDS system. All MDS CC's are of the form

$[(Y\ y) | P(@\ y)],$

to be read as "'The collection of all instances, y, of the template Y, such that the PREDICATE, P(@ y) is satisfied'" [Irwin & Srinivasan 1975, p. 36]. The at-sign refers to the "current instance" of the template whose CC this is (i.e., the particular thing currently being described when this CC is invoked). Predicates are in a simple logical form, and may call functions defined in the MDS language (these are defined as "\$F" templates). The forms tested in CC's are mostly relational triples (x r y), meaning "x appears in relation r to y". Fig. 8.7 [Irwin & Srinivasan 1975, p. 74] illustrates the body of CC10, accessed from the

CC10-
((CAUSEDESN C) | (@ causes C)
 (C causedesnof:statedesn @)))
STATEDESN causes)

Figure 8.7. An MDS consistency condition.

STATEDESN template (see Fig. 8.1). In Fig. 8.1, the anchor (STATEDESN causes) calls for the slot to be filled with a CAUSEDESN, which is defined elsewhere to be a set of CAUSEDESNs (the "\$L" in Fig 8.1 specifies that the called template is to be a "list template"). Here we see that for a given CAUSEDESN, C, to be a member of that set for a particular STATEDESN, S, C has to be pointed to by a "causes" link from S ("@" means S here) and C has to point to some thing which points with a "statedesn" pointer back to S.

There is also a small set of pre-defined system functions which are used in the CC's. Consistency conditions add power to the simple use of

a template as a VALUE/RESTRICTION, by allowing arbitrary first-order constraints to be placed on slot fillers. Cross-slot dependencies can be embodied in the CC's, since they can access any information available on a relational path from the "current instance". But, as in the frame system, mutual dependencies must be expressed at both anchors*.

We should note here that the procedural language forms we have seen get their use primarily as further constraints on role fillers. None of the systems allow these constraints to be defined in the same language as that in which their conceptual structures are expressed, and therefore, they cannot be considered explicit knowledge of the system (they may be considered to be known, but implicitly, since they can only be evaluated). This is the problem that I have tried to overcome by expressing structural conditions in the same notation as concepts. I will examine a bit further the implications in Section 8.4.

8.1.3. Some special forms

As I noted in Section 7.3, paths through a network are difficult to represent in terms of nodes and links. I was forced to extend the mechanism to include multi-headed pointers -- these were embodied in substructure reference nodes. In contrast, a relational path is very easy to express in a lexical notation, since what we might want is "the X (slot) of the y (frame) which fills the z (slot) of the q (frame) which ...". Each of the three notations that I have been discussing takes advantage of this relative ease of path description. For example, we might have in MDS "(@ causalnetof: causalmodel: causalnetdefn: terminalstate: iinstance S)", or in KRL we might see "the husband from a Marriage with wife = the secondWife inUnit Henry". An interesting feature of FRL is the use of indirection, indicated by "=>". We might

* This is also the case in KRL.

have

```
(MEETING (TIME ($PREFER
              ((=>IRA (MEETING TIME) $PREFER))
              ((=>RBR (MEETING TIME) $PREFER))))),
```

which references the "meeting time" slots in the frames IRA and RBR. This is the special meaning of non-atomic slot names, like "when duration" in Fig. 8.4.

A feature of the graphic notation evolved in this report is its ease in pointing to any single place. Uses were found in Chapters 6 and 7 for pointers from role nodes not only to concept nodes outside a given concept, but to other role description nodes within the same concept, as well as to the enclosing concept node itself. These latter kinds of reflexive reference are perspicuous and easy with lines and arrows. With lexical notations, however, special considerations must be made, since we cannot "point" back to the enclosing structure. KRL-1 provides the language form "my" to refer to the contents of a slot of the unit being described. Thus a Marriage unit might have its self slot containing "a Relationship with participants = {my husband, my wife}", where "husband" and "wife" are other slots in the unit. In KRL-0, the form "ThisOne" allowed one to point back to a prototype that appeared in the SELF slot (e.g., "(the hometown from Person ThisOne)"), and the SELF slot itself is a place to talk about the current unit as a whole. As we have seen, MDS provides the form "@" to indicate the "current instance", i.e., the particular thing being described by a template. This form is used in CC's, which are associated with the anchors of the template. (Anchors, again, are pairs like (STATEDESN causes), which we might consider to be a dattr of STATEDESN.) Finally, FRL provides the global variables ":frame", ":slot", and ":value" to refer to "this frame", "this slot", and "this value". Other slots of the current frame can be referred to by their names, with "!" indicating that the value of the specified slot is what is desired.

One final commonality in the syntax of these three languages bears

noting: slot names cannot be duplicated within conceptual units. In addition, combinations of superconcepts with the same slot names can cause problems (this is the reason that KRL perspectives have a single prototype, and all perspectives in a description, while describing the same entity, are treated independently). Taking a closer look, we notice that there are no intervening system constructs that might allow multiple slots with the same name -- the only notation for distinguishing the slots is by name. Looking back at SI-Net notation, we see that this is not a problem (although a lexical surface notation for our representation would have to have a convention for distinguishing between dattr's). Each role node is independent of the others, and has an explicit ROLE link to indicate its role. The node itself is a structure and not just an atomic "name" (and the DATTRS link is a system primitive). Thus, as we saw with HYDROGEN/BOMBS (Fig. 5.1), we might want three nodes named "FUEL" -- i.e., three FUEL slots. These each have independent identity in the graphic notation, since pointers can go directly to the role nodes (also briefly discussed were possible lexical referencing conventions). In addition, while I have not discussed multiple superconcepts, explicit pointers to role nodes (the ROLE links) could keep separate all inherited dattr's.

8.2. Basic representational structures

The discussion of language forms could not have proceeded completely independent of the underlying representational objects of KRL, MDS, and FRL, but it has not covered in much depth the important aspects of units, templates, and frames. In this section, I delve into the structures one builds with the language forms illustrated in the last, and look at some important associated notions like individuation and inheritance -- that is, I shall look at the "syntax" of the representations themselves.

8.2.1. Structured conceptual objects

Each of the systems we are considering has made the move away from representing lists of independent "facts" or relational triples. Instead, knowledge is "chunked" into groups of descriptions of closely associated entities. Thus, the structured concept nodes we have constructed in earlier chapters correspond fairly closely to KRL units, MDS templates, and FRL frames. Each is composed of a set of role descriptions, further characterized by what we might call "role facets", features of the individuals which will fill the role. Facets that I have introduced include NUMBER, MODALITY, ROLE, and VALUE/RESTRICTION*. Fig. 8.8 gives a schematic characterization.

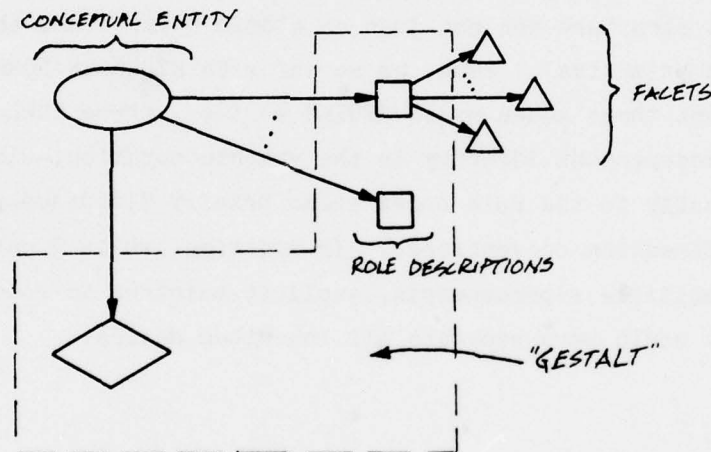


Figure 8.8. Generalized "chunk" structure.

We should bear this general structure in mind as we proceed through descriptions of its correspondents in each system. From the diagram, we can conclude that a knowledge representation of this form must account

* This is not quite precise: the notion of a "datrr" captures information about the potential fillers as individuals (V/R), and as a set (NUMBER, MODALITY). It also serves as a placeholder for the definition of the functional role itself, and therefore "ROLE" is not a facet in the above sense.

for at least the following:

- 1) the joining together, at the conceptual entity, of all role descriptions and structural "gestalt";
- 2) the definition of the relations linking each piece to the entity as a whole -- in our notation, these are DATTRS, DMODS, DINSTS, DIFFS, and STRUCTURE;
- 3) the joining together, at each role description structure, of its associated facets;
- 4) the definitions of the facets themselves -- in SI-Nets we considered the facets VALUE/RESTRICTION, NUMBER, and MODALITY;
- 5) a structuring interrelationship that explains how the role descriptions fit together (i.e., relationships between roles and between roles and the whole);
- 6) relationships between different conceptual entities (i.e., non-structuring relationships); and
- 7) relationships between conceptual entities and others connected to them in special relationships (i.e., INDIVIDUATES and DSUPERC) -- this involves not only links between "chunks", but correspondences between parts ("inheritance" relationships, e.g., ROLE links combined with the information in DATTRS, DMODS, DIFFS, and DINSTS*.

In SI-Net notation, I have tried to account for each of these important points. All of the above kinds of relationships in our nets were made to be explicit -- as we shall now see, many of these important links are only implicit in some of the other formalisms. This makes the SI-Net notation well-suited for a discussion of the characteristics of the others. First, I will consider the conceptual units and their structures. Then I will look at more global relationships (points 5

* This brings to light the fact that these epistemological relations are not as clean as I would prefer. DIFFS and DMODS point to the same type of node as DATTRS, but change the inheritance import of ROLE. Therefore, the two imports should probably be separated into DATTRS and DINSTS from concept to role node, and ROLE-MOD, ROLE-DIFF, and ROLE-INST between role nodes. See [Brachman 1978].

through 7).

The KRL unit has a set of slots (created by the user) which contain descriptors. Descriptors generally embody a class restriction for values (i.e., the VALUE/RESTRICTION facet), and the perspective-type descriptor points off to a partially (or fully) filled-in unit which describes the potential filler of the slot. Particular values can take the place of perspectives to indicate that the slot is "filled". KRL-0 had "individual" units to correspond to instances, but KRL-1 leaves the interpretation of a unit as a pattern or individual to its interpreter. Slots, then, always have descriptions, really, and not "particular values". KRL slots correspond to our set of dattr's, but do not have aspects other than VALUE/RESTRICTION. Thus, other role facets like NUMBER and MODALITY are not accounted for by the KRL representational system.

It is possible to add "footnotes" in KRL-1 to slots, and therefore achieve the effect of additional constraints on slot fillers. These are called "meta-descriptions", and are often used, for example, to indicate that the descriptor in the slot is a default to be used when no particular slot filler is known. This is a way to add facets to role descriptions, but particular facets are not built into the notation.

MDS templates are like KRL units, but have a more relational flavor. The set of slots is indicated by triples of the form (<relation> <template> <inverse-relation>), where the <template> is the value class restriction for this slot. (The relations are really names for the role descriptions (dattr's), as were the KRL slot names.) Any <relation> link added will cause the inverse link to be added automatically, so that all relations are two-way. The generality here leaves the template feeling less like a chunk of knowledge with a "self" than one of the old semantic network nodes with a simple set of links to an arbitrary set of other nodes.

CC's augment the simple value class restriction that MDS allows (in

Section 8.2.1
Structured conceptual objects

a relational statement, only a single template can be named), and in these would be embodied the notion of NUMBER. Modality seems not to be an issue in MDS -- all attributes are necessary to form an individuator. Individuators in the system are placed in the special "instance" ("immediate instance") relation with their template definitions, and are merely sets of relations with other individuators, one for each relation defined in the template.

One additional MDS feature is the set of "flags" associated with the relational definitions. These flags can specify the type of an individuator, for example, so that, as in STATEDES, an individuator can itself be a template (the "TN" flag means "terminal node" -- individuators are themselves node templates). The "C" flag on a relation symbol indicates constant -- the template following the relation symbol in a triple (the value class restriction) is passed intact to any individuator (this is like a DINSTS pointer to a role node). These flags add to the definitional power of the simple triple structure.

Frames, as I mentioned earlier, are merely lists of slots. Slots in FRL do have facets, which are lists of keys plus properties. These facets are arbitrary -- to achieve a NUMBER restriction, an explicit uniqueness predicate must be included somewhere in the slot by the user, and he must assume that it gets evaluated at the right time. FRL provides some conventions, including the indication by the "\$value" key of a particular value "filling" the slot, the "\$require" key providing predicates that must be true of fillers, facets for procedure invocation at particular times, like \$if-added and \$if-removed, and facets for aid when trying to fill slots, like \$if-needed and \$default. As we mentioned, the language syntax does not enforce these conventions. Filled frames, by the way, are no different from general ones. The opposite of the "ako" ("a kind of") link is called "instance", which points to a frame which in turn can have instance links.

As we saw in Section 8.1, procedures can be attached at various places in these notations. In FRL, any key can have LISP procedure calls as properties. MDS has only CC's, which are expressed in a special sublanguage (and TR's -- transformation rules; these are like \$if-needed methods and are associated with slots). These are always associated with slots. KRL has "demons" and "servants", again, expressed in LISP, with possible reference to KRL structures. In fact, procedures in all three languages can reference arbitrary pieces of structure, and tend to be used as additional filler constraints. They generally are accessed through slots of the defining concepts.

While all of the representations maintain role descriptions in the form of slots, with at least some number of facets, none really separate out a structuring interrelationship for those slots. As I mentioned, the CC's of MDS can access any items on a relational path from the anchoring individuator, and therefore, within a CC one can express a relationship between "this slot" and some other slot in the individuator. One can probably express a relationship between "this slot" and the thing as a whole, although the template as an entity with a "self" does not play a large part in the MDS methodology. To reiterate, CC's are associated with role descriptions (i.e., MDS "anchors"), and are generally used to express special constraints for their own role fillers. There is some use made of slot interdependencies, but not in any systematic way. All knowledge of global interrelations is implicit in the CC's, although some parts of the MDS system can "read" these CC's and draw some conclusions about them. However, in general, interdependence is implicit in the evaluation of the CC's.

FRL has much the same kind of facility for implicit structural knowledge. Since LISP procedures can be interspersed among role facets, and are arbitrary, they can express knowledge about virtually any kind of relationship between conceptual entities. Fig. 8.5 illustrated the use of this kind of relationship for slot mutual dependencies. Note

that these procedures contain only implicit knowledge -- they can be evaluated, but not examined. Again, they must be associated with individual slots. We could, perhaps, create a "structural-condition" slot for a frame, and keep all interdependence information in it. However, no discipline exists for such a use, nor would the procedures even then be introspectable.

KRL seems not to be concerned at all with slot interdependencies. The self slot would probably be the right place for structural knowledge, and as I mentioned in the discussion of reflexives, references can be made to other slots in the unit in KRL itself (e.g., "a Relationship with participants = {my husband, my wife}"). In addition, the meta-description facility might be used to attach structural relationships to the unit as a whole. As with the other systems, however, no precedent exists for this type of use*.

In sum, we can say that all three representations embody some, but not all, aspects of the basic structure of Fig. 8.8 for conceptual entities. Each has slots, with at least some minimal differentiation of facets (all have VALUE/RESTRICTION facets, none have NUMBER or MODALITY, and all allow procedures of arbitrary effect to be invoked). None of the three separates relationships between role descriptions and conceptual units in the way that I have advocated (i.e., making slots independent of their names), and thus all look like the older semantic

* There exists a further problem here: if an interdependence is mutual, then if not expressed in a single central place, it should be expressed at both slots that it affects. But the natural KRL expression of this fails to capture the fact that there is a single mutual constraint. For example, consider a unit with a slot defined as follows: "h: the husband from a Marriage with wife = my w". A w slot in the same unit might then read "w: the wife from a Marriage with husband = my h". Note, then, that this implies that there are two Marriages, rather than a single one "with wife = my w" and "husband = my h". The KRL-1 solution is to use a footnote on the first mention of Marriage (in the h slot), and to have the w slot read, "w: the wife from my noteref 1". This at least works, although it is not particularly perspicuous.

nets in terms of immediacy of relations -- this makes it hard to distinguish actual internal slots from arbitrary relations with other concepts. In addition, the burden is on the user in FRL to maintain two-way links. All believe in an open-ended set of roles, unlike linguistic case theories and Schank's conceptual dependency, and thus conform to the ideas on cases expressed in this report. Each believes also in procedural attachment -- to slots -- and none (except perhaps for MDS) has yet a method for understanding the procedures. Finally, while each has the rudimentary facility, no one of the three representations uses a structuring interrelationship over all its roles.

8.2.2. Individuation structures

KRL, MDS, and FRL all use their basic conceptual entities as patterns which implicitly describe classes of other entities. Those other entities which fit the descriptions are in each of the systems referred to as "instances" (I have here chosen to call the descriptions, "individuators"). Recall that in the earlier discussion of the intensional nature of concepts we determined these to be "individual concepts" -- intensional entities describing individuals in the domain. I also discussed a useful subconcept mechanism, which allows conceptual descriptions to be restricted in their role descriptions or structural condition. Here I discuss the corresponding entities in the three knowledge representation languages.

MDS is the only one of the three to insist on an absolute difference between an individuator and a conceptual pattern (template). There is a simple, one-level operation of individuation, which adds an "instance" [sic] relation between defining template and individuator. The individuation is complete when all relation symbols in a template are assigned values -- these values are themselves individutors of objects in the data base. There is no notion of subconcept in the system.

However, individuator can themselves be templates, as in the STATEDESN case we encountered above (see Fig. 8.1 -- the "TN" flag indicates that the individuator is itself to be a template). If the "C" flag is indicated, the template specified by the relation is passed intact to the individuator. For example, in Fig. 8.1, the relation "status" has a "C" flag -- any individuator of STATEDESN (the template in which this is embedded) will have a "status" relation to STATUS (the particular template indicated in STATEDESN), not to an individuator of STATUS. This way a requirement can be passed to a lower concept, such that an instance of that concept would satisfy the requirement (fill the slot). This can be used as a rudimentary subclassification device, but slot requirements cannot be modified.

Neither KRL nor FRL draw a firm distinction between general conceptual entities and individual concepts. The older KRL-0 differentiated individual units from others, but that kind of unit typing is gone in KRL-1. Instead the interpretation of a node as an individuator or a generic concept depends on the state of the interpreter. While a very flexible mechanism, this means that the underlying primitives do not exist for unambiguously representing an individual. Thus, one cannot really distinguish the filling of slots from the description of potential legal fillers. For example, we could not distinguish the underlying types of description in "Aaron is a traveller" (he satisfies the predicate) and "a sailor is a traveller" (the two predicates have an intensional relationship).

There are no links between general concepts and their individuator in KRL except for the "self" slot in the individuator. In units generally taken to be individuals, the self slot is the only one present -- it indicates a set of descriptions that the current unit satisfies. Thus the units in KRL strongly separate the descriptions that describe the current unit (i.e., those in the self slot) from descriptions of instances of the current unit (i.e., those embodied by the unit's other slots). Thus, as we see illustrated schematically in Fig. 8.9, KRL

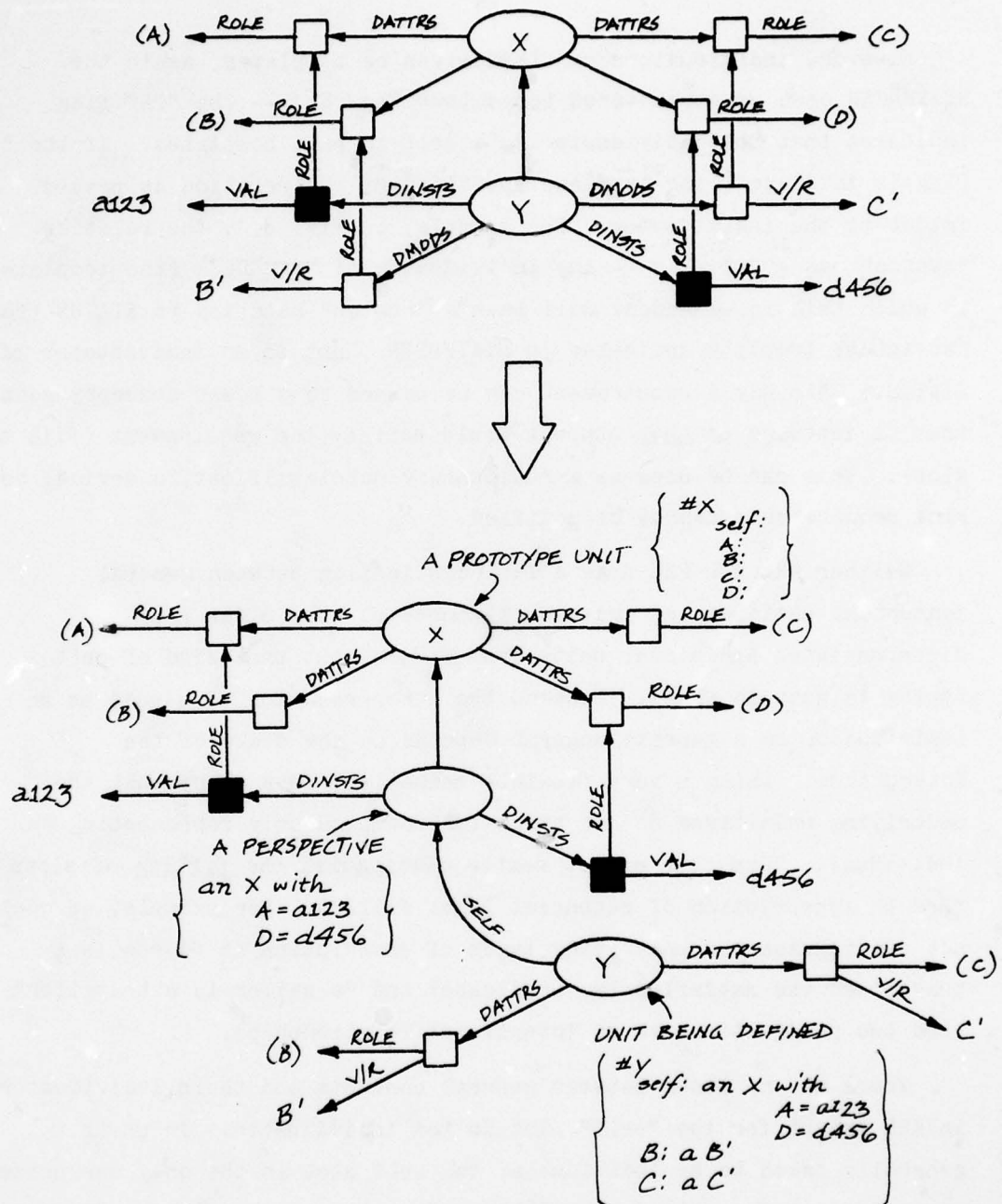


Figure 8.9. A closer look at properties in KRL.

forces apart properties coming from nodes "above" from those to be passed to nodes "below". There is no type of "ROLE" pointer in this notation to explicitly tie these types of properties together (this is

true of the other two representations as well). To indicate dependencies of new slots on those passed from above, we need to construct references through the self slot.

FRL has no special "self" slot (although one can be defined and used), and only when a \$value property is added to a frame do the constraints on the slot make the frame act like an individuator. The inverse of the "ako" link is called "instance", giving us some indication that the FRL authors do not choose to make an explicit differentiation between individuators and subconcepts. Subconcepts can exist by virtue of the implicit inheritance of slots of the same name from all concepts accessible from the ako slot. There is no representational mechanism for indicating explicitly the relationship between a slot and others of the same name in higher frames (i.e., no differentiation between DMODS and DIFFS). The discipline for lining up slots is the same as the one employed by earlier semantic nets -- names are repeated. Fortunately, the differentiation between \$value and \$require facets disambiguates the meanings of the slot names.

8.2.3. Inheritance

The reason that subconcept formation is not natural in KRL is that system's peculiar type of inheritance. There seems to be a single-level approach to the passing of slots in KRL -- a perspective inherits all and only the slots from its own prototype. Notice that it is not a unit that inherits slots, but a perspective, which may be used only as a subpart of a unit. Thus, something which is a type of Traveller is not of necessity considered to have the same slots as the Traveller unit does. While it may have a perspective in the self slot which says "a Traveller with ...", its own slots are completely independent of those of Traveller -- the information in the perspective is local to that perspective only.

This is so because of KRL's basic orientation toward multiple perspectives. If all slots were inherited directly by the unit, then confusions would arise when different prototypes had slots of the same name (I commented upon this earlier -- the one presented in this report is the only notation with enough explicit links to differentiate between non-identical slots with the same name). Therefore, it appears that rather than attempt a general-to-more-specific progression of subconcepts in KRL, one would instead use multiple perspectives to describe each of the aspects of the target individual. Any constraints that need be "inherited" or modified in these multiple perspectives would have to be passed by reflexive accesses in slots. For example, a Doctor unit might have the following two slots:

self: a Professional with field = my specialty
specialty: a BranchOfMedicine

This structure essentially creates a Doctor subconcept of Professional, with the "field" dattr VALUE/RESTRICTION being restricted to a BranchOfMedicine (this would be indicated in SI-Net notation with a DMODS link -- here, the KRL syntax equates the "field" role of Professional with the "specialty" role of Doctor).

Thus we see that to have constraints passed more than one level, explicit reference must be constructed. On the other hand, the one level move from prototype to perspective apparently comes for free with the system: "...properties of the prototype individual are assumed true of the individual being described unless explicitly counterindicated" [Bobrow & Winograd 1977, p.8]. Two kinds of properties are relevant to this inheritance -- the non-specific descriptors used as VALUE/RESTRICTION indicators are passed intact, and apparently can be considered "properties associated with the prototype"; and default assumptions can be explicitly indicated. This latter type of property is essentially instructions to the pattern-matcher of KRL to assume a given value if none other can be found, and is specified in KRL-1 by meta-description:

waitPerson: a¹ Waitress 1: a Default.

Thus, default properties are explicitly specified (i.e., "a Default" is explicitly indicated), but implicitly inherited (the perspective slot would not really be filled with the value)*.

The same local type of inheritance we see in KRL is present in MDS. Requirements specified in a template (i.e., the called template as well as CC's) must be satisfied in individuators of the template, and no other requirements are relevant. The special case of the "constant" flag allows a requirement to be passed down a level intact, so that an individuator of an individuator will be forced to satisfy it. Thus all inheritance operations are simple, explicit, and unambiguous -- in individuators all and only those constraints local to the defining template are satisfied, except for those specified by the "C" flag one level higher, which are inherited (unmodified). The individuator will have an explicit relationship to each slot filler (or inherited constraint).

The CC's in MDS are allowed arbitrary access through all relation paths leading from an individuator. Therefore, in a sense, properties of more general (or any related) concepts can be considered to be "inherited" by the individuator -- they can be used in CC's just as local properties can. However, these properties would be used in CC's but not really inherited as explicit parts of the individuator. In addition, the notion of "default" is not relevant in the MDS methodology -- an individuator either fills a relation with an individual in the data base or inherits explicitly the particular value called in the defining template.

* Defaults are an interesting and problematic feature of many of the current representation languages. Smith [1978] claims that they are meta-descriptive information -- instructions to the interpreter -- and are therefore not at the same descriptive level as say, VALUE/RESTRICTIONS. I concur with his analysis.

FRL has a broader type of inheritance, much like that embodied in our network notation. The ako links define an inheritance hierarchy such that all slots of frames located along an ako path with the same name are considered to be part of the same role. Thus frames further down the path will specify further constraints on the slot fillers; facets are taken conjunctively such that the \$require facet of the "X" slot of some frame would include (as far as processing is concerned) not only its own property, but the \$require properties of the X slot of the frame of which it is "a kind of", and all \$require properties of X slots of parents of that frame.

Values are inherited in the same way, except that the routines which determine inheritance usually take the first \$value property found rather than the conjunction of all \$value's in the ako path. However, the comment facility allows the routines to continue looking for \$value's (and conjoin them) if so desired. Thus, inheritance of properties and constraints is implicit in the notation (there are no ROLE pointers to link together groups of role descriptions).

FRL has a "\$default" key that is used to specify default properties for particular slots, and this feature constitutes the same kind of signal to the matching routines as its KRL equivalent.

8.3. Representation semantics

The representation scheme developed in Chapters 4 and 5 was motivated by an urgent need for an explicit semantics for semantic network-type formalisms. We were led to an examination of the operations underlying nodes and links, and eventually tried to make each of the epistemologically primitive relationships an explicit link in SI-Nets. Thus, the SI-Net language is rather barren in terms of user-level niceties -- the formalism is really itself only a "semantics" for networks. I intentionally restricted attention to basic knowledge-

structuring principles, thus producing a neutral notation upon which many different kinds of domains could be constructed.

This accent on epistemological primitives as a language for expressing knowledge structures separates this work from the other three projects. While it is probable that each of these notations could be represented in terms of the other, none of the others would provide much perspicuity about the underlying operations of role description, modification, individuation, and inheritance, or about the type of structuring interrelations devised here. This is because they are languages built on top of implicit semantics (since they each have a well-defined set of programs to process structures, each does have, in some sense, an underlying "semantics"). To understand the worldviews underlying KRL units, MDS templates, and FRL frames, we need to perform the same kind of semantic analysis that we did earlier with semantic networks.

I will first consider some general aspects of these representations, and then focus in on the interpretations of particular structures.

The basic KRL emphasis is on description -- units implicitly represent classes of objects by grouping sets of descriptions that those objects can satisfy. This is very close to my own interpretation of semantic net concept nodes, and therefore KRL can be expected to have many of the intensional mechanisms that SI-Nets embody. In addition, KRL explicitly acknowledges that all descriptions are inherently partial, and is built around that fact. MDS has a more relational flavor. Objects are classed by the relations in which they participate with other objects. This has much the same feel as the earlier semantic network notions that I have mentioned, although MDS, with its CC's, begins to make considerable improvement on the older methodology. FRL seems to emphasize the hierarchical nature of descriptions of classes, again similar to older semantic nets. This too, however, represents a significant extension over the older net notations, with the allowance for slot facets and the integration of procedures.

Recall that one of the problems we encountered with semantic nets was the direct encoding of domain-specific information in the primitive notation (with, for example, links supposedly representing high-level conceptual relationships). I advocated, instead, the intervention of a clear and well-specified foundation of representational primitives, out of which to build these higher-level concepts. If the conceptual domain is defined in terms of well-understood primitives, operations like individuation are automatically defined for all future extensions of the network. The KRL, MDS, and FRL notations stack up pretty well in this respect, since by using the structures defined by the system authors, the user ends up with well-formed underlying structures for his concepts (i.e., one cannot build ill-defined structures). However, FRL is a little weak on this point, since the user can design domain-specific facets, and therefore is required to write routines in LISP, which is not constrained by FRL, to implement their "semantics". If one uses only the system-defined facets, however, FRL will provide the semantics. None of the three systems seems to be representationally complete. The main problem is quantification -- multiple fillers of a role, and quantification over them, are not considerations in any of these languages (although sets are included, and a quantification mechanism may be coming in KRL-1). In addition, hypotheticals, mass terms, and lambda-abstraction are missing (these are missing in our notation as well).

Another pervading theme in earlier chapters of this report was the making explicit of all underlying operations, so that any particular representational structure could be unambiguously interpreted from the meanings of primitives. MDS seems to follow this philosophy, although its representational repertoire does not contain relations like DATTRS, DMODS, ROLE, etc. However, the semantics of individuation is so simple, nothing is hidden. The structure that Srinivasan presents as underlying his templates and "instances" [Srinivasan 1976] has everything directly connected to everything else it relates to. KRL does an excellent job

of separating part-whole relationships from whole-whole relationships. The self slot explicitly separates the relationships in which the unit participates in as a whole from the "cases", as illustrated above. As I pointed out, however, this makes for a peculiar type of concept specialization, since requirements cannot be passed down directly. Instead, a perspective in the self slot will reference other slots in the unit. Unfortunately, the interpretation of the unit as a whole (individual or template) depends "on the state of the interpreter", and thus the difference between the value of a slot filler and the description of one (as in the modifying of a requirement) cannot be reflected unambiguously in the notation. At once, depending on "how you look at it", the description in a slot can be thought of as filling the slot and as describing fillers-to-be. This makes the semantics of a unit ambiguous.

The facets in FRL allow one to keep most aspects of knowledge structure explicit (as for example, \$value and \$require separate use and mention of requirements), but as I mentioned, there is a broad type of inheritance in the representation that makes much implicit. In addition (and this is also a problem with KRL), the intervention of arbitrary LISP procedures causes much knowledge to be hidden from the representation.

In all fairness, it should be here noted that authors of each of these systems are currently looking into the representation of procedural knowledge in their notations, and that a formal semantics for KRL, called KRS, is currently being worked out [Smith 1977]. MDS has its semantics formally described in [Srinivasan 1976]; and FRL is not as ambitious a project as the other two.

8.3.1. Meanings of basic structures

The basic structures of each of these systems -- templates, units, and frames -- all implicitly represent classes of entities in the domain being represented. However, as was the case with our own intensionally-oriented concepts, the explicit focus is on description of an individual entity by specifying each of the parts such an individual must have. Thus the representations are all object-centered.

The simplest of these major units -- the MDS template -- adheres to a strictly definitional philosophy. Every instance must fit its defining template's requirements exactly, in all aspects. Thus, an instance is defined by exactly the set of relationships specified in the template.

KRL, on the other hand, espouses a more descriptive approach. A unit is a set of descriptions, not necessarily "complete". The KRL authors contrast this approach with the more standard definitional one: "There would be no simple sense in which the system contained a 'definition' of the object..." [Bobrow & Winograd 1977, p.7]. Since units are not considered to be definitions, they are more like descriptions of stereotypical individuals -- "typical" members of the classes implicitly represented by the units. Thus the descriptions that general units embody may not apply directly to any particular individuals, but they do capture the "general idea" of members of the classes*.

I should add here that the impression one gets is that the use of a unit as a stereotypical individual is only relevant in an explicit "description by comparison" -- a "perspective". When the interpreter so

* Constructing the meaning of a class from a growing set of descriptions of its members seems to reflect more the way that humans form concepts -- but we should not rule out the possibility that we abstract out definitional "rules" once we are confident of class membership.

desires, the unit will instead represent a particular individual (which is not unrealistic, since descriptions can be made by comparison with a particular individual as well as a standard prototype). Or, when creating an individuator of some particular unit, the unit will serve as a definition. The KRL-0 paper states that a prototype ". . . combines [all of] the default knowledge applied to members of the class in the absence of specific information" [Bobrow & Winograd 1977, p.8]. While this is undeniably true, I claim that there is a much stronger definitional sense to the knowledge embodied in the prototype unit. When constructing a perspective, the only slots that are allowed to be further specified are the ones that are defined in the prototype unit. Thus the prototype in KRL is a very constraining definition for all its perspectives (which we can consider its "individuator").

Thus we have three types of logical entity which a unit can at the same time represent -- a definition, a stereotype, and an individual. This makes for a confusing semantics, since there is no way to explicitly separate these types of units if we so desire. KRL has captured an important aspect of the reasoning process here, however, in the realization that the same thing can be viewed in many ways, depending on the intent. I believe that the semantics should account explicitly for the different types of view, but that the ambiguous unit interpretation should also be maintained, and defined in terms of those more primitive meanings for units. I have attempted to do this in SI-Net notation, for example, with the different types of nominalization links for different senses of a verb that share essentially the same structure.

One final note about KRL units -- these conceptual units are the only ones in the three representation languages to differ significantly from the basic structure of Fig. 8.8. As mentioned earlier, the self slot separates information inherited from above in a generalization hierarchy from those constraints to be passed on to units below. Thus,

The diagram illustrates a semantic network with the following structure:

- Nodes:**
 - U1, U2, U3:** Represented by rectangles.
 - P:** Represented by an oval.
 - Intermediate Nodes:** Several unlabeled rectangles represent intermediate nodes in the network.
- Edges and Labels:**
 - U1 to P:** Labeled "SELF".
 - U2 to P:** Labeled "INDIVIDUATES".
 - U3 to P:** Labeled "SELF".
 - U1 to Intermediate Nodes:** Three edges labeled "DATRS" lead to nodes with "ROLE NAME" labels, specifically "slotName1", "slotName2", and "slotName3".
 - U2 to Intermediate Nodes:** Two edges labeled "DATRS" lead to nodes with "ROLE" labels, specifically "(ROLE1)" and "(ROLE2)".
 - U3 to Intermediate Nodes:** Two edges labeled "DATRS" lead to nodes with "ROLE NAME" labels, specifically "ROLE3" and "ROLE4".
 - Other Edges:**
 - From U2 to a node labeled "X" (labeled "VAL") and to a node labeled "Y" (labeled "ROLE").
 - From U3 to a node labeled "X" (labeled "VAL") and to a node labeled "Y" (labeled "ROLE").

further inheritance below the perspective P (although not all dattrs of U2 are necessarily instantiated). The upper SELF link points to a perspective P which stands for "a U2 with ROLE1 = X ROLE2 = Y". The lower self link represents a pointer to a KRL-1 slot perspective, "the ROLE3 from a U3".

* Which, recall, I postulated to be more like cables than simple links.

provided that the user does not create new internal structures. But since the structure is an "FLIST" -- a list of lists of lists -- what the user can make of it is open-ended. Frames are claimed only to be a useful data structuring technique, with a set of system functions for things like ako-inheritance, procedural attachment, etc. The preferred system view is much like that of KRL, with frames standing for individuals or prototypes or strict definitions under varying circumstances. However, the user can augment these semantics merely by adding LISP code to the system.

In both KRL and MDS, the basic units can be explicitly specified to be functions which return a value (the standard interpretation is more relational, with individuators being propositions -- see Section 5.3). KRL-1 includes the concept of a "functional" -- a function-like abbreviation for a slot perspective. For example, "ChildOf(Aaron, Rachel)" is a syntactic abbreviation for "the child from a Marriage with father = Aaron mother = Rachel)." MDS uses the "\$F" flag to make a function template -- such a template uses the "FNDEF" relation to capture a CC or CC-like structure and return its value as the result. Frames apparently do not work like functions; LISP is used instead.

8.3.2. Role descriptions and structure

The philosophy of representation developed in this report advocates constructing concepts out of interrelated role descriptions. A special primitive link, DATTRS, was created to circumscribe the set of roles and keep them distinct from the structural condition. The functional role played by each part was indicated explicitly by a ROLE link (thus allowing two distinct dattrs to play essentially the same functional part in the overall complex). Role descriptions had some important facets, whose task it was to characterize the potential fillers of the role -- ultimately, the particular filler in an individuator was indicated with the VAL link. Relationships in which the role filler was

to participate were included in the structural condition. Thus were segregated descriptions of role fillers as entities from descriptions of the roles they played with respect to their enclosing concepts (i.e., the structural condition embodies the definitions of the roles by capturing relations between their fillers and fillers of other role descriptions, the fillers of the roles and the whole, and fillers of the role descriptions and outside concepts).

Each of the other systems captures at least part but not all of this philosophy in its slot definitions. As I have pointed out, none of KRL, MDS, or FRL separates the functional role from the slot, and thus unique slot names must be used. Each does have a mechanism for constraining the class of entities that are potential fillers of a slot, but none has a NUMBER or MODALITY facet, nor a role differentiation capability built into the representation.

MDS has the consistency condition facility based in its slots -- CC's provide extra constraints on slot fillers, since the basic notation allows only the naming of a template as the VALUE/RESTRICTION. MDS authors like to think of the CC as embodying the "semantics" of the anchoring relationship. KRL slots are used to describe "substructures" which are deemed significant for comparison purposes. They are a set of closely associated descriptions to be chunked around a type of entity -- these can be parts of the entity, or other important aspects. In a perspective, the slot fillers are thought of as "further specifications" of aspects of the prototype. This interpretation supports my interpretation of perspectives as concepts, with properties not specified being inherited, and those specified being conjoined with those from the defining prototype. FRL slots include certain procedural facets which provide implicit semantics for the slots, in much the way the MDS CC's do. The facet facility, as pointed out, is completely general, and facets can be specified by the user. The \$value, \$require, etc., facets are helpful features provided by the system, and in themselves are a useful generalization of the MDS and KRL single-faceted

(VALUE/RESTRICTION) slots.

None of the systems has anything to say about a structural "gestalt" for the concepts as a whole. All further constraints, including procedural ones, are considered to be part of the slot/role descriptions and not as general definitions for functional roles within a complex.

8.3.4. Individuation and individuator

The meanings of individuator and inheritance in these systems should be clear from our discussions above -- individuator are considered individuals only in MDS, while the KRL and FRL interpreters can at times construe prototypes to be individuals. Only MDS, then, really has an unambiguous representational semantics supporting individuation. As I have mentioned, we might consider the perspective in KRL a better candidate than the unit for this type of operation, but there is still no explicit distinction between individuals and templates. FRL has an individuation facility which tries to construct \$value facets for all inherited slots, but there is no exclusivity for these facets -- the notation does not dismiss the possibility of \$value, \$require, and \$default existing side by side in a slot. The semantics of a frame with such a slot is not clear.

In all cases, general concepts implicitly define classes, so that what individuator there are, are considered to represent class members. An interesting development in KRL-1 is the introduction of "coreference descriptors". Such references are an attempt to pick out "individual" as the interpretation of the unit pointed to. For instance, if Aaron is "a Person with sex = Male", we do not mean Aaron has a sex "whichIs Male" (and is currently not known), but that "Male" is the value of that attribute. This is an interesting way of making up for the lack of an explicit "individual" representation, and in fact, may be a more useful interpretation of individuality.

AD-A056 524

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MASS
A STRUCTURAL PARADIGM FOR REPRESENTING KNOWLEDGE.(U)
MAY 78 R J BRACHMAN
BBN-3605

F/G 5/6

N00014-77-C-0371

NL

UNCLASSIFIED

4 OF 4

AD
AD 66524



END
DATE
FILMED
9-78

DDC

One important commonality among KRL, MDS, and FRL is "instantiation" to execute processes that are implicitly general concepts. One can accomplish the scheduling of a diagnosis of a disease by asking for a SCHEDULE frame or template to be individuated. All of the constraints embodied in frames and templates, and especially the \$if-needed TR's in MDS), will cause the step-by-step production of a final state that the interpreter must achieve -- explicit characteristics can even be specified within the general in MDS, a general problem-solver can be invoked to determine next to achieve the final state. This type of individuation is an important contribution of all of these methodologies.

8.4. Conclusions

The three methodologies that I have been investigating form a cluster of ideas about knowledge representation that is consistent. This current Zeitgeist differs from older ideas about knowledge representation mainly in its insistence on structured conceptual entities, and the direct part played by processes. It is generally a declarative, object-centered framework. Structure is achieved by clustering around a single conceptual complex (unit, template, frame) a set of unordered slots. The complex then serves as a pattern to be followed in particular cases as the slots are filled. We can consider the underlying process to be the association with a type of entity of a set of roles by "pieces" of that entity, and the filling of slots to be the assignment of particular players to the roles they take on in the whole.

Thus, the slot has two important jobs: 1) the description of the class of entities which might legally take on the role described by the slot; and 2) the description of what that role is with respect to the other roles in the complex. Procedures can be incorporated into the complex both to allow procedural checking of constraints on fillers (i.e., to embody implicit class description criteria) and to capture (implicitly) the role to be played by an individual piece or group of pieces.

This compellingly consistent image of a structured entity uncovers an aspect of knowledge structure that is lacking in the three systems from which we have culled the image. These systems have to some extent descriptions of many types of entities by virtue of descriptions of their parts. But they have only very shallow (at best) descriptions of how those parts go together to make the group into a "whole". For example, it is easy to make a frame for, say, "communication" -- we make it a-kind-of "activity", and specify its participants. But what makes it a communication? What happens during such an activity? The criticism of case-like structures in Chapter 5 still holds. A description of the participants and the end state of an activity is no explanation of that activity.

As mentioned earlier, none of the current representations allow us to express procedures in the same language as descriptions of objects. KRL, MDS, and FRL have all made giant leaps over case notations, in that they at least allow the inclusion of executable procedures which can implicitly describe what makes a whole more than the sum of its parts (although it is not clear that this use is generally made of the facilities*). Not until the "gestalt" which holds all the pieces

* One might guess that the KRL "self" slot is the place for a description of the roles (rather than the role-players). However, as I have shown, the self slot is really only a distributed DSUPERC/ako-type of link.

together is expressed in the same kind of introspectable terms as units, templates, and frames will these knowledge structures be complete.

8.4.1. SI-Net notation as a pedagogical tool

It has been relatively easy to express features of these other representations in terms of our own network formalism. The SI-Net seems to capture explicitly the fundamental knowledge representation operations that underlie these fairly elaborate, user-level notations. I have explicitly tried in my own efforts to separate out each primitive knowledge representation aspect, and as a result, have accounted for virtually all of the issues derived from this study of KRL, MDS, and FRL. This includes not only the distinction between role and role-player description, but also the explanation of the structural gestalt of a conceptual entity in a language compatible with the description of the entity and its pieces. I have recognized that the structural condition is a distinct, different type of entity than the slots in a concept, and have tried to illustrate its place in my epistemology of concepts.

As I have mentioned, the graphic notation is really closer to a semantics than a user-level representation language. All of the other systems rather freely intermix procedures defined in other languages with "pure representation". Not only is LISP code embedded, but instructions to matchers, property inheritance routines, and concept builders are allowed in many places. Much domain-specific information exists at the same level as FRL primitives. Thus the nature of the knowledge representations themselves is a bit hard to sort out. I have here tried to distill what really goes on in these representations, taking into account not only the explicit structures but those implicit in these contacts with extra-representational mechanisms.

In sum, it appears that FRL is a simple uniform language for

Section 8.4.1
SI-Net notation as a pedagogical tool

capturing structured concepts. However, the structures that it supports are limited, because of the lack of a structural condition-like mechanism for relating the parts. Much is up to the user, since very few representational primitives are fixed. In this regard, FRL resembles the older semantic nets, where everything was just a node with unconstrained links. On the other hand, FRL allows you to facet its slots, and add comments on values, and it also allows you to intermix LISP code in your frames (however, it is not clear how much representational power this buys, since procedures are associated with slots only). The particular paradigm for procedural attachment offered by the authors of FRL (\$if-needed, \$if-added, and \$if-removed) is not general enough to allow specification of complex, quantified relations between slots. Ako-inheritance is built in and is a powerful uniform mechanism -- if you choose to use it. However, it is not differentiated enough to provide the flexibility of the "decentralized" SI-Net inheritance links (DMODS, DIFFS, and DINSTS, in conjunction with ROLE), which provide inheritance on an individual role basis. All told, it is clear that FRL exerts very little worldview on its user -- it is a system with little epistemology built in by its authors. It is more a useful data structure with a package of useful routines available if you want them. Everything is a frame, and beyond that, it's up to you. You even have to maintain two-way links yourself.

MDS is also a very general uniform language. There is a bit more epistemology in the representation than in FRL, causing the user to think in terms of slots with single classes of fillers. The slots can have CC's; and more generally, templates have instances. But this system, too, is a bit like the old semantic nets in its uniformity -- everything must be a template, with relational links. This homogeneity is limiting in its definitional power, as we saw in Chapters 4 and 5, and in particular prevents the construction of subconcepts and different template types for things like hypotheticals, lambda-abstractions, and mass terms. The language allows more relational path following than the

older nets, and CC's provide much more power, but there is not much help available to guide you in breaking your particular domain into the right pieces. While the semantics of MDS are clear and unambiguous, the language is shallow, and it is not clear how easy it is to use.

KRL is a much more complex representational language than the other two. It is geared to the kind of descriptive task that I have spoken of at length in this report, and the authors have thus thought about intensional issues and incorporated them into a strong epistemology. Obviously, this is a more ambitious effort than FRL, and is not geared to problem-solving as MDS is. While the representations of these other two resemble the older semantic nets in the simple uniformity, KRL more closely resembles our own SI-Net structure in its attempt to capture the basic knowledge representation primitives in its semantics. While these semantics are not perfectly clear at the moment (there is the ambiguity we discussed earlier), and the structure of roles is not accounted for, these will probably be coming soon*.

* See [Smith 1978] for an in-depth analysis of some of the more subtle aspects of the semantics of representation languages like those studied in this chapter.

Chapter 9. Conclusions

In Chapters 1 and 3, I sketched a picture of two consulting tasks that I felt would be useful for a computer to perform. The chapters that followed presented and analyzed a structural paradigm for representing the knowledge of a consulting program. The time has now come to assess how close to the realization of the ultimate goal this representation has brought us. First, I will discuss briefly some of the important general contributions of the Structured Inheritance paradigm. Then, in Section 9.2, I will discuss the representation in the context of the consulting task, spending some time on how the representation could be used in that environment. Finally, I conclude with the future -- what needs to be done to make the representation better, and a sketch of some projects for which this thesis has laid the foundation.

9.1. Dattrs and structural conditions

The Structured Inheritance formalism developed in this report exhibits some important characteristics not available in other networks. For example, I have introduced the notion of a "dattr" of a concept -- a closely associated part or attribute, along with the context for that part. Dattr's, I would like to emphasize, are not just parts, but any features of an entity that can be considered criterial to its definition. As Woods [1975a] has pointed out, previous notations did not generally distinguish between such attributes and "arbitrary relations".

I discussed how dattrrs were structured entities, with at least a value class restriction, a modality, a number specification, and a functional role to be played by the filler. Many semantic network creators overlooked this complexity in resorting to attribute/value pairs -- it was thought that by simply labelling a link with a role the proper structure could be expressed. As I illustrated in Chapter 4, this led to ambiguities in the attribute links, and ignored things like optional and multiple fillers of the same role.

Further, the older nets implicitly expressed the belief that attribute links (like COLOR, etc.) were defined by nodes, at which such information could be stored. Unfortunately no substantive use was made of this assumed facility, and the problem was glossed over. In contrast, I have in this paper tried to emphasize the importance of roles in the description of concepts. I have shown how the attempt to incorporate Fillmore's "case" notation into the foundational level of semantic nets misses the point: there are not a small number of universal primitive roles exhibited by objects in relation to actions. There are, rather, similarities between many roles, but almost always local variants -- idiosyncratic interpretations of roles depending on the particular concepts in which they are defined. This observation is possible only because the structural condition defines the meanings of the roles.

The structural condition is another significant contribution of this type of representation. It provides the meanings for the roles associated with its defining concept by expressing a set of relationships in which fillers of those roles are to participate. Since it allows this expression in terms of all other concepts in the network, no small set of knowledge primitives or canonical structures is necessary. This reflects our beliefs on how human memory is associative and circular (like, as we have pointed out, dictionary definitions are). In addition, if we provided some minimal structural conditions (as in, e.g., "the HYDROGEN is somehow related to the BOMB"), we could build

level upon level of "vaguely-defined" concepts, all dependent at the bottom on the trivial structural condition. This way we can reflect the lack of depth of many of our personal definitions of complex concepts like HYDROGEN/BOMB.

Taken together, roles and structural conditions give us a precise way to define "concept", not as a set of features, but as a set of role (dattr) descriptions and a structuring interrelationship that describes how potential fillers of the roles are to interact.

At a more global level, perhaps the most important feature of the SI-Net formalism is the level at which it expresses relationships. Rather than encode conceptual information directly into uniform nodes and links, I have chosen to provide a fixed set of node and link types that express relationships between concepts as formal objects in a representation. The set of epistemological primitives that make up the representation allows precise expression of the relationships underlying a particular fact, event, or object. In addition, "well-formed concepts" are defined by the syntax of the formalism, since we insist on a fixed structure for each node type. Generic concepts and individuators can be constructed automatically, since the linkage to represent them is defined in advance (see Section 9.2.1.1). The notion of an "epistemology" is notably absent from earlier semantic network languages, and as we have seen, it is thus difficult in many languages to ensure consistent interpretation of links.

9.2. The structure of a consultant's knowledge

My primary intent with SI-Nets was to provide a language in which we can express a consultant's knowledge of his area of expertise. In the Introduction, I considered a set of requirements for knowledge structures to be used for consulting about documents and consulting

about Hermes. Later, in attempting to construct a knowledge base for each of these domains, we saw how the representation faced each of these issues. Let me here summarize how the formalism presented in this report handles the important requirements of the consulting task:

- Foundations -- as I discussed in Chapters 4 and 5, older network notations were inadequate in several respects at the fundamental level. Links were used ambiguously, logical operations of the representation were confused with knowledge of the domain being represented, and there was no account of the structure that held the arguments of predicative concepts together. The approach here was to try to make it possible to represent all aspects of the consultant's domain unambiguously, including nuances if they could be distinguished. I argued that this required what I called an "epistemology" -- a separate level of representation that explicitly accounted for all and only the operations on the representation; concepts were to be constructed out of "epistemologically primitive" links, and no relation of the domain itself was to be encoded as a link*. As a result, the foundational problems of older semantic nets do not arise in this notation; each underlying representational operation is explicitly available as a single piece of the notation. The syntax of node types determines how well-formed concepts are to be constructed.

With the exception of some unconsidered details (which I will mention in Section 9.3.1), the Structured Inheritance Network provides an adequate representational foundation on which document topics and Hermes objects and commands can be constructed.

- The representation of structured objects -- the nominal compounds that we encounter as document topics, and the objects manipulated by Hermes must be considered to be structured entities; as mentioned above, previous notations have not provided more than a skeletal account of the internal structure of concepts. SI-Nets allow the explicit representation of the relationships that can exist between the parts and closely

* With no domain concepts as "primitives", the question arises as to how a program using this type of knowledge structure might "get started" -- how it would relate conceptual knowledge of the domain with its low-level perceptual mechanisms. In Section 9.3.1, I discuss briefly a primitive type of structural condition which would account for the basic domain-dependent concepts ("knowledge primitives") needed to start the otherwise circular definitional mechanism.

associated attributes of a structured concept. The structural condition expresses how the roles of an object are tied together, in terms of other concepts available in the knowledge base. This type of structuring avoids the problems of "case" notations (which rely on a small set of so-called "universal" relationships), and as we saw in Chapters 6 and 7, is adequate to represent the structured entities of the two consulting domains.

- Deriving new concepts from old -- I utilized the notion of intension (Chapter 5), and showed how the representation expressed the intensions of natural language designators like predicates, functors, sentences, and individual expressions. This allowed us to specify precisely the meanings of our links, and gave us a set of definitional relationships between concepts. The primitive links expressing binding (DINSTS, ROLE, VAL) provide a precisely-defined individuation facility; since the individuation mechanism is defined in terms of primitive links only, it is always clear how to derive an individuator from a concept (even a new, unanticipated one). By the same token, the modification links (DMODS, DIFFS) provide an unambiguous mechanism for forming subconcepts, again well-defined for all concepts, existing or potential.

As Chapter 7 illustrated, the modification and individuation mechanisms account for a taxonomic hierarchy of Hermes concepts and the ability to reflect a user's current program environment. The fact that the links reflecting modification and individuation are fixed and unambiguous provides a way to infer certain relationships not represented explicitly -- although I did not present such an inference mechanism in this report (see Section 9.2.1.2). In addition, the clear way to form new concepts from existing ones should allow a program to easily assimilate certain kinds of new information -- I also return to this below (Section 9.2.1.1).

- Relating nominal and verbal concepts -- the SI-Net paradigm provides a mechanism for deriving certain types of nominalizations from verbal concepts. The nominalization links provided all of the groundwork for the representation of nominal compounds involving verbal elements. In addition, the structural condition gives us a way to represent how actions operate on objects, an important aspect of the Hermes program domain. SI-Nets represent nominal concepts as structured objects, as well as the verbal ones handled by previous notations.
- The representation of idiosyncratic interpretations -- by allowing the structural condition to define the relationship between

roles in terms of arbitrary combinations of other available concepts, I have provided a mechanism for expressing idiosyncratic interpretations of concepts (i.e., there is no need to insist on a "canonical" interpretation in terms of a set of predetermined knowledge primitives). This is especially important in the document consulting domain, where the understanding of references and particular topics varies with experience and exposure to other parts of the literature. We have seen how this facility can account for many different levels of sophistication in the representation of a concept.

- Paraphrase retrieval -- I have not explicitly covered how SI-Nets can account for paraphrase. However, at least one way to make use of the paradigm is through the definition mechanisms that it provides. If A is defined in terms of B, then it should be possible to recognize a discussion of A couched in terms of B (e.g., "a transcribing command used for summarizing messages" should be easily recognized as "a summarizing command"). Further, other characterizations of B could be used -- since all definitions are explicit in this notation, there are a great many ways to arrive at a node by following a descriptive path. I discuss this in more detail in Section 9.2.1.2.

The above synopsis underlines how this report has focused on the representation of the domain, and has consequently covered only a small class of operations on the representation. In the next section, I will sketch some ways in which the representation might be used by a consulting program.

9.2.1. Using the representation

The SI-Net formalism is a possible way to structure the knowledge that a consulting program would have about its area of expertise. Since the structure is a declarative representation of the domain, it must be accompanied by a set of routines which operate on it. I envision consulting programs as interactive tools that can assimilate new knowledge and be queried in natural language about things they are expected to know (including inferred concepts). This type of behavior would entail a natural language processor of some sort, that could parse both assertions (i.e., statements to be assimilated) and queries into

some semantic interpretation compatible with the network structure. An ATN parser [Burton 1976] and Woods' "FOR" notation ([1968]; see also Section 5.1.4) are possible candidates for the processor and query language.

Assuming that an appropriate natural language "front end" could be constructed, there are some important functions to be provided before the knowledge base can be put to intelligent use by the consultant. Here I briefly discuss how the representation provided herein supports three of these: assimilation, paraphrase, and inference.

9.2.1.1. Assimilation of new information

In both the document consultant and the Hermes on-line assistant, an extensive initial phase of learning will be necessary. The important basic concepts for understanding document topics must be encoded before the document consultant can begin assimilating particular topics; and the structure of the Hermes program must be encoded before the assistant can be expected to answer questions. Once this "bootstrapping" is accomplished, both systems will be called upon periodically to incorporate still further knowledge, in the form of new topics and particular user sessions*. Therefore, the ability to assimilate new knowledge is critical to either consulting program.

The representation in this report supports a certain amount of assimilation automatically. The structure of concepts is well-defined in advance; in this scheme, each concept can be altered in only a small set of ways, viz. for each dattr of the concept, a restriction (DMODS), differentiation (DIFFS), or particularization (DINSTS) is possible. In each of these three cases, the particular modifications that can be made

* The Hermes assistant could conceivably be asked, in addition, to alter its picture of the program when Hermes itself changes.

are constrained by the currently existing set of concepts. That is, if the VALUE/RESTRICTION for a dattr of concept C is X, then subconcepts can be formed from C by restricting the dattr to subconcepts of X. Alternatively, in the case of DINSTS, the dattr can only be particularized to some existing individuator of X. This means that in a finite data base, the number and structure of the subconcepts that can be formed below any concept is determined at the time the subconcept is to be formed. In another view [Woods, personal communication], there is an "implicit lattice of potential concepts" that exists below any concept in the net. Each dattr of a concept can be restricted by all of the subconcepts of its VALUE/RESTRICTION -- if we imagine a set of subconcepts formed by such a chain of restrictions, we have one "dimension" of an implicit lattice. If all other dattrs were similarly restricted, and all combinations of those restrictions formed, we would have an imaginary lattice describing all concepts that can be defined using the rules of the notation and the starting concept. Assimilation in many cases, then, becomes a matter of merely finding the place for a new concept in this lattice. Any concept incorporated in this way can be said to be assimilated*, in the sense that it is immediately available for interpretation of further new concepts, since its relationships to all other concepts in the net are accounted for.

Another important feature of the formalism that supports assimilation is the way that the structural condition is defined in terms of existing concepts. Thus, an ARCH could be defined as a THING with a LINTEL (BRICK) and 2 UPRIGHTs (BRICKs), where

FOR EVERY x / UPRIGHT ; SUPPORT(x, LINTEL)

* This includes under one heading both "assimilation" and "accommodation" of concepts found to fit into the lattice. Accommodation of the existing knowledge structure to new discoveries and generalizations is more difficult, and I have not considered wholesale changes to the structure of concepts that might be caused by the assimilation of new information.

and

```
FOR EVERY x / UPRIGHT ;  
  (FOR SOME y / UPRIGHT : NOT(EQUAL(y,x)) ;  
    NOT( TOUCH(x, y))) .
```

If the definitions of the SUPPORT and TOUCH relationships existed in the network, this definition could be assimilated and used immediately for inference and individuation.

9.2.1.2. Paraphrase and inference

Since definitions in SI-Nets are couched in terms of other concepts, a set of useful paraphrase relations is available at each concept. A retrieval request might be expressed by referring to the superconcept of the desired concept, or by describing the set of relations expressed in its structural condition. It should be possible to follow definitional links to the target concept from those in which the request was worded.

The key to the ability to understand at least one class of paraphrases is the fact that nodes are named not only by their print-names, but by their definitional derivation from other nodes (their "EGOs"). For example, among other things, the SURVEY/COMMAND is a SUMMARIZING/COMMAND whose DEFAULT SEQUENCE/ARGUMENT is CSEQUENCE. This definition is directly derivable from the links of Fig. 7.10. Since all individuator of a subconcept are also individuator of its superconcept, and all roles are inherited by the subconcept unless specifically blocked, a paraphrase is easily derivable from the structure in this way: follow the DSUPER link from SURVEY/COMMAND to SUMMARIZING/COMMAND -- this transition represents the "is a" phrase. The DATTRS link from that node to the role description for SEQUENCE/ARGUMENT represents "whose", and a similar transition can be made for the VALUE/RESTRICTION of that role; the two roles placed together give us "DEFAULT SEQUENCE/ARGUMENT". Finally, the DINSTS-VAL pair represents the statement that the DEFAULT role is filled by

CSEQUENCE (i.e., the DEFAULT "is" CSEQUENCE). Because all definitional links are explicit, these paths are available to a processing program.

In addition to paraphrase, an operation that is critical to the utility of the formalism as a consultant's knowledge structure is inference. Along with "potential concepts" and inherited properties, features of unanticipated situations should be derivable from this type of knowledge base. It is one thing to represent the structure of Hermes statically so that simple questions about its properties might be answered; such a task does not demand the meticulous detail of the structures of Chapter 7, and would probably even benefit by shallower, more vague definitions. It is quite another to be able to handle unanticipated circumstances and "understand" what the program will do. The following example, taken from an actual Hermes-related experience, will serve to illustrate what I mean by "unanticipated" circumstances -- those which the network was not planned in advance to handle:

Consider the following hypothetical situation: a Hermes user has always used the LIST command to print his messages on the lineprinter, and has always defaulted the template used for printing; thus he has never had to type more than one argument (LIST takes three -- a sequence of messages, a template for formatting, and a file as destination). However, he decides this time to list his messages on a file, and types this command:

(9.1) >LIST ALL STANDARD NEW.MSGS
(this means to LIST ALL messages in the current file onto file NEW.MSGS, using template STANDARD to format them). When he eventually prints the new file on the lineprinter, he discovers that the messages are not separated by page breaks, as they normally are. His response is, naturally, "Why didn't my messages get separated?" (The actual answer in Hermes is that the default template, LTEMPLATE, used when a template is not explicitly specified by the user, contains the item, SEPARATE, whereas the explicitly given template in (9.1), STANDARD, does not.)

Let us look briefly at what it might take to answer such a question from the type of structure presented in Chapter 7. If we can create the desired structure using only the paradigm presented in this report, then, provided that the program can be written to process the representation, an on-line assistant might conceivably answer this question.

Section 9.2.1.2
Paraphrase and inference

First, let us assume that we have a smart language processor, so that the interpretation of this question is expanded to the equivalent of "Why didn't my messages get separated when I typed 'LIST ALL STANDARD NEW.MSGS', whereas they always do when I type 'LIST ALL'?" The query passed to the inference component will have to reflect this question. I will not here worry about its details, except to note that it must demand a comparison between two EFFECTs -- the EFFECT of "LIST ALL" and the EFFECT of (9.1) -- with a request for the reason why messages were not separated in the latter case. That is, it will tell the program to create two command invocations (i.e., individuators of LIST/COMMAND), and to look for individuators of an effect that I will call "SEPARATE/MESSAGES".

Considering the EFFECT structures of Section 7.3, it is not unreasonable to postulate an effect for separating messages -- SEPARATE/MESSAGES. This effect would place a special character in the output stream to cause a page break. The main use for this effect is in the routine, TRANSFORM/MSG/THRU/TEMPLATE, discussed in Section 7.3 and illustrated in Fig. 7.11. In Hermes, if a "SEPARATE" template item is a part of the template taken as argument to TRANSFORM/..., the SEPARATE/MESSAGES routine is invoked. The SI-Net representation would reflect this fact by including in the structural condition of TRANSFORM/... a paraindivuator of SEPARATE/MESSAGES embedded in a call to some kind of conditional. While I have not previously provided an EFFECT for the conditional, this EFFECT representation should be feasible with just the representational apparatus of Chapter 7, and a sketch of its structure appears in Fig. 9.1. So the inference component is to create two LIST/COMMAND individuators, and investigate their EFFECTs for SEPARATE/MESSAGES calls.

The desired structures are achieved in the framework of Chapter 7 by creating two individuators of LIST/COMMAND; each is initialized by instantiating the SYNTAX dattr with the expression typed. As mentioned in Section 7.4, the structural condition of LIST would express the

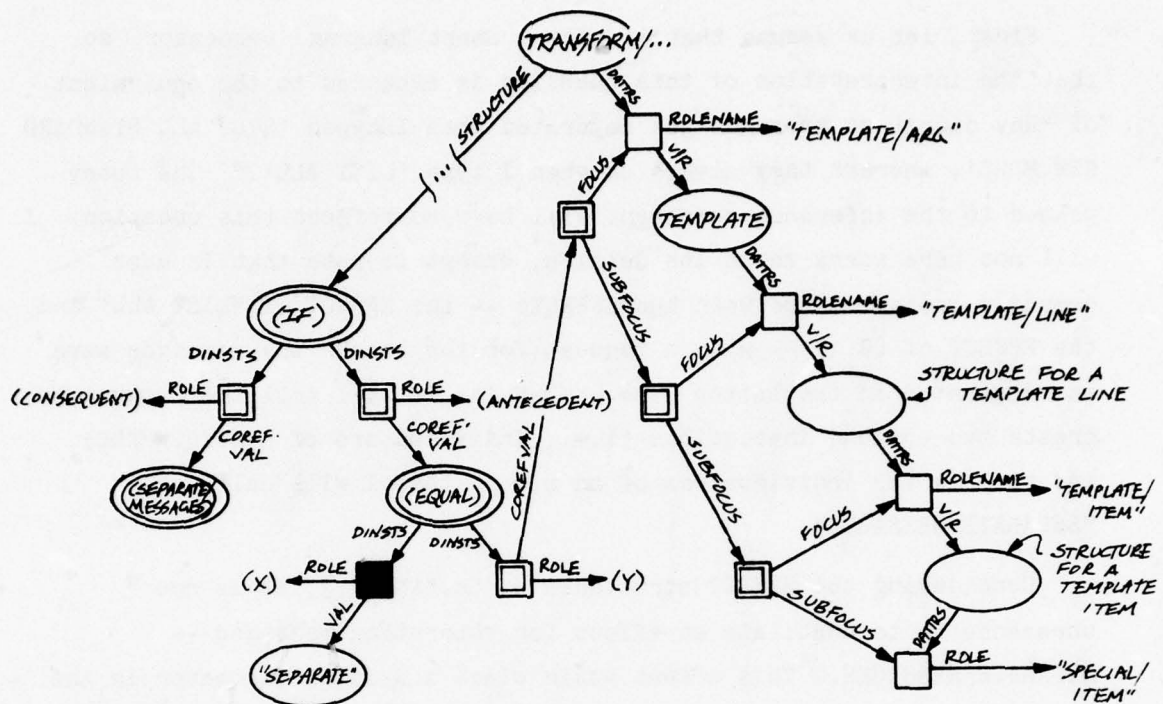


Figure 9.1. Part of the S/C of TRANSFORM/MSG/THRU/TEMPLATE.

relationship between the SYNTAX role and the other roles. Using that, then, we could derive the ARGUMENTs in each case; in the "LIST ALL" individuator, two of the arguments would have to be defaulted: the TEMPLATE would be LTEMPLATE (a special default template used for listing) and the DESTINATION would be the lineprinter (file LPT:). In the (9.1) case, all ARGUMENTs can be derived explicitly from the typed names.

Given the ARGUMENTs, the particular EFFECT of each command individuator is determined. Again, the structural condition, as we saw in Section 7.3, expresses the relationships between the VALUES of the ARGUMENTs of a command and its EFFECT. Therefore an EFFECT individuator can be automatically created once the ARGUMENTs are individuated. Both of the EFFECTs produced here would at some point make a call on an EFFECT introduced in Chapter 7 called "TRANSFORM/MSG/THRU/TEMPLATE" as their basic operation.

The critical ingredient in the understanding of the problem is the elaboration of Fig. 7.11 that is sketched in Fig. 9.1. What is needed is an expression of the precise mechanism by which a message gets transformed using a template (i.e., the structural condition of TRANSFORM/MSG/THRU/TEMPLATE -- "for each item in the template, retrieve that part of the particular message . . .", etc.). Somewhere in the definition of TRANSFORM/... would be a call on SEPARATE/MESSAGES; this call would be made in an individuator only if the driving template had the item, "SEPARATE", as one of its parts. The connection between the presence of this item in the template and the execution of SEPARATE/MESSAGES is the piece of structure that would allow an inference program to complete this piece of reasoning. This connection is provided by the (IF) token in the structural condition in Fig. 9.1.

The EFFECTs individuated to represent the two command invocations would be examined statically (i.e., without their being "run"), in which case it would be confirmed that in one case, the TRANSFORM/... call will cause separation, while in the other it would not. The "reason" that the SEPARATE/MESSAGES branch of TRANSFORM/... would be called in one case, but not in the other, is that one of the templates (LTEMPLATE) has a "SEPARATE" item in it, while the other (STANDARD) has not. It should be possible to infer this reason automatically by observing how each of the two templates individuates the TEMPLATE concept (see Fig. 7.16). In one case, the dattr for the SPECIAL/ITEM role has a filler (it is pointed to by a role instance node), while in the other, it is left unfilled.

The user can be informed of the answer that STANDARD has no SEPARATE, while LTEMPLATE does (the request was for why SEPARATE/MESSAGES was invoked in one case and not in the other). The information about LTEMPLATE will be irrelevant, however, unless the user is told that it is used as the default value, since he never typed its name. And, since Hermes supplied the value, the user probably should be informed. This appears to be good policy in general, since the effects

of commands are significantly affected by their arguments, many of which are normally defaulted. (In general, defaults are a source of confusion to many users.) In this case, the answer really called for is something like, "STANDARD is not the default template for LIST, so you do not get the same effect." However, one would not expect to have the program able to infer that the user probably typed "STANDARD" because he thought it was the default he normally got with "LIST ALL".

While this discussion is strictly speculative, the only type of representation that it have relied on is that presented in Chapter 7. Thus, provided that we can write the program to build and analyze the structure, we could conceivably achieve such a complex inference.

9.3. General use and specific shortcomings

Before discussing the specific things that remain to be done, I discuss briefly the general use of SI-Net formalisms. In addition to its possibility as a memory structure for a consulting program, the representation has the potential to be used as a general pedagogical tool for explaining concepts. In it are embodied explicit entities for many of the kinds of abstractions we might want to talk about in dealing with concepts, relationships, facts, and events. There also exists a primitive link for each fundamental relationship that can exist between these, so that we can unambiguously represent the details and nuances of the concepts of a particular domain (see Chapters 6 and 7 for examples)*. Shallower representations like the older semantic nets

* I offer the analyses of nominal compounds and the Hermes program as important contributions in their own right to the study of knowledge representation. Dattr helped us break nominal compounds into groups with consistent underlying structure. We saw that despite the complexity and often seemingly arbitrary nature of Lees' [1963] syntactic classification, there are only two major kinds of relationships underlying these compounds, namely, concept plus dattr, and dattr plus dattr. In the Hermes structure, there is produced a

cannot represent subtle distinctions in meaning, forcing the human reader of the notation to infer missing details. SI-Nets, in contrast, allow the expression of more shades of meaning by forcing the representation to be unambiguous and by offering an adequate set of epistemological primitives. Thus, SI-Net formalisms can be used to explain the underpinnings of others, as witnessed by Section 4.3.4 and Chapter 8.

As a result of its formality and well-specified semantics, this representation is also directly implementable on a computer. Subject to a few implementation decisions*, a program to build and use these networks could easily be constructed from the specification we have given (a program is being built which allows the user to build networks using the epistemology).

On the other hand, the insistence on explicitness and detail makes this a difficult language for writing down concepts. The figures in this paper are so complex as to disallow any claim to "intuitive obviousness". In addition, they are time-consuming to draw. If we desired to build a large data base with many definitions (for example, if we wished to encode the entire Hermes program), it would be tedious to be forced to account for each role description node and its set of links -- this is no doubt a motivation for the simpler "attribute/value" notation. What is needed is a habitable surface notation that allows detail to be left out when desired and offers the easy encoding of

detailed picture of the important parts of a complex program and, if we had a full editing program that would allow the definition of new concepts in terms of existing ones, the rest of the program could be similarly encoded. The description in Chapter 7 was given in a form suitable for direct computer implementation. Covered in a declarative form were both commands and objects, and the complex relationships between them.

* For example, it should be determined whether or not to represent explicitly all dattrrs at a node, or force the DSUPERC chain to be searched each time one needs to be accessed, etc.

complex concepts*.

While I do not here offer such a surface notation for conversing with the computer about concepts, that one should be possible is suggested by the following feature. The concept-building and modification mechanisms are defined in terms of a fixed set of primitive links. Since the syntax for concept and role nodes is fixed by these links, the details of the linkage could be supplied automatically by a program. Thus the user could be freed from dealing with DINSTS, INDIVIDUATES, ROLE, etc. links, and could simply say to a program, "I want to individuate (define, modify) a concept." For any concept that exists (or can exist) in the network, the way to individuate it is defined by its set of dattr, so the program need merely look at all DATTRS, DIFFS and DMODS links, and request, for example, "I need 2 SUPPORTs for this individual; please designate 2 BRICKs." An experimental program has been constructed that in fact insulates the user from all primitive links, and exhibits this type of prompting behavior. This program, however is not yet a complete editor, nor at the moment does it constitute a language in which concepts can be written down.

* One might suggest that a language similar to that of KRL is more appropriate for this task. While KRL is undeniably a good surface notation, its semantics is not explicit and does not cover the broad range of nuances and structures that ours does (see Chapter 8) -- we still need a "universal" notation so that all combinations of concepts and relationships can be expressed in it. What is required is to work from the semantics out, rather than from the surface language in, and develop a KRL-like lexical language that corresponds to our representation.

9.3.1. What remains to be done

While I believe that the Structured Inheritance Network represents a significant advance over previous semantic nets, it is not yet completely specified. Further work must be done on several aspects of the representation.

One of the most glaring deficiencies in the representation is the lack of procedures incorporated directly in the notation. Certain types of dattrrs would probably benefit from links directly from role description nodes to primitive routines that could be invoked 1) to determine whether or not the potential filler was legal (a procedural VALUE/RESTRICTION), 2) to fill the role on demand (i.e., like the \$if-needed method of FRL), and 3) to determine actively the number of fillers required. More importantly, the structural condition needs to access procedures both for checking applicability and for constructing an entity out of a set of parts.

The first case has several aspects. While I have intended structural conditions as predicates to be checked against a set of potential role fillers, some predicates are in practice impossible to check. For example, consider the concept, MARRIED*. The precise definition of being married involves the occurrence of a particular ceremony at some past point in time, and no following occurrence of a particular kind of legal action (divorce or annulment). Unfortunately, past points in time are not directly available to our own perceptual apparatus, nor would they be to a machine's (we cannot prove that two people are married in this definition). All we have to go on are "traces" of various sorts (legal records, rings, etc.). To a precise-minded machine, these would be insufficient to apply the MARRIED

* Bill Woods has suggested this as a representative of a class of predicates whose applicability may be impossible to determine procedurally from observable data [personal communication].

predicate. What we might propose then, in addition to the definitional structural condition (which is still needed for inference purposes) is an alternative structural heuristic that will allow us to determine whether we are justified in concluding that the structural condition is satisfied.

Another use for a procedure here would be to allow a series of quick gross checks that, if violated, would mean that the structural condition could not possibly apply. This would save a lot of effort for many kinds of processing, since the full structural condition would be applied only when it was close to being guaranteed to succeed. Finally, the nature of network representations assures us that there will always be a class of concepts that are not defined in terms of others. These "knowledge primitives" (the set of which is not determined by the epistemology, but, rather, by the user) should have procedural structural conditions to determine their applicability. Procedures directly accessible here would reflect our own direct correlation of certain predicates with perceptual mechanisms (e.g., "red", "sweet", etc.).

While I have accounted for most of the important aspects of nominal compounds and the Hermes program, there are still many aspects of knowledge representation left indeterminate by the formalism. I have neglected to treat "mass" concepts and continuity, and have provided no explanation for "quantification" over a mass term like "hydrogen" in the HYDROGEN/BOMB example (Section 5.1). It is not clear from the small set of structural condition nodes how to express complex quantifications over infinitely repeatable dattrs. I am also not sure that the simple notation offered in passing for a SET (Fig. 4.1) is adequate, nor have I accounted for an ordering mechanism when dattrs have multiple fillers (it is not clear whether something primitive in the notation is required). The representation is also not yet equipped to handle hypotheticals -- an explicit assertion of existence should be added to free individuator of implicit existential import.

Further, the important problem of relative clauses needs to be investigated more fully. The SI-Net representation presented here provides an explicit handle on a role filler in context -- role description and instance nodes give us an intermediary, a place to talk about role fillers in a given context without making statements about them in general. This may be the key to the resolution of several problems with the representation of relative clauses discussed by Woods [1975a, pp. 60-65]. Since we can select the context as well as the participant, we can distinguish in the underlying representation between the embedded sentence and the matrix sentence of which it is a part. This would be accomplished by selecting the role node that captures the role to be played in the constituent sentence by the relativized participant. The constituent sentence would be accessed from the matrix sentence through one of its dattr's and thus the two sentences would be represented asymmetrically. As a result, this kind of representation should not exhibit the problems with symmetric relations discussed by Woods.

9.4. Three follow-up projects

There are three further important investigations that I believe should be pursued. First, a full-scale implementation of the ideas in this paper would prove invaluable in measuring the practical utility and inference capability of the representation. As I mentioned, a "habitable surface language" should be developed for the graphical representation. A set of interactive tools for editing networks would follow; ultimately a graphical interface, at least to display the endproduct, would be an ideal facility. The visual import of the type of illustration that we have produced in this document makes this a particularly desirable tool. This type of implementation effort is currently being undertaken at BBN.

Second, the detailed structure of roles and their interrelationships should be energetically pursued. Roles are emerging as perhaps the most important aspect of conceptual representation, and as I hinted in Chapter 5, they probably themselves have a hierarchy of structure. How the structural condition patterns that make up the role definitions can be abstracted and related is an important new direction for research.

Finally, a feature of the notation that we might call "decentralized inheritance" deserves future attention. In explicitly breaking out relationships like DMODS, DIFFS, and DINSTS, I have shifted the designation of how properties are to be inherited away from the single inter-concept link (DSUPERC, INDIVIDUATES) to the role links themselves. As a result, any combination of inheritance and instantiation relations for dattr can be expressed explicitly through the dattr-link counterparts; in previous network structures, either all had to be inherited (by subconcepts) or all had to be instantiated (by individuators). Clearly, the older view was oversimplified, since for any concept, each dattr can be restricted, differentiated, or particularized. A virtually endless supply of inter-concept links ("cables") would be necessary to express all possible combinations of operations on dattr in a centralized inter-concept link. In addition, since each role node points with an explicit link back to its defining role, it is not clear that the inter-concept link is even necessary at all! This is a fairly radical view, and should be pursued with caution, but the implications of decentralized inheritance should now be investigated.

Appendix Historical Perspective on Nominalizations

Appendix: Historical Perspective on Nominalizations

Lees' [1963] account represents a great step in the understanding of nominalization. In this Appendix, I consider several of the nominal types mentioned by Lees that I consider important to the representational task of Chapter 6. Two of these nominals, the Action and Gerundive nominals, play important parts in many compounds. I first discuss the ideas introduced by Lees, and then examine two variant opinions, offered by Chomsky [1970] and Fraser [1970].

A.1. Lees

Agentive nominals are names for agents of actions. They usually occur with the "-er" morpheme, and the nominals thus formed are always concrete. There is no debate over the structure of the Agentive nominal, and, as illustrated in Section 6.3.2, it is relatively easy to find a place in the SI-Net conceptual framework for nouns like "lover", "drummer", "owner", "maker", "fighter", etc. When an Agentive of the -Er variety is found as the head of a compound, it is easy to determine its relationship to a preceding nominal modifier, since it is the agent of the action named, which in turn is done to, for, or with the attributive modifier (e.g., the head of "bull fighter" names the agent of the action, "fight", whose object is "bull").

The Factive nominal has several forms, but all have to do with the conversion of a sentence into a fact. An event can be described in a declarative sentence, and from that event a Factive nominal can be derived that expresses the fact that the event happened (information rather than the activity itself). Lees proposes two forms, a "that-" clause and a question-word ("wh-") clause, and illustrates how both

forms may occur as the (abstract) subject in sentences in which the verb is copulative:

(that-form)	That he came was obvious.
(wh- form)	What he did was obvious.

Notice in these sample sentences how the events discussed are dealt with as wholes -- as facts -- and no concern with the activities themselves is shown.

The other contexts for Lees' Factives are the following: as subject of one of a special class of verbs that take animate objects ("That he came astonished them," "What he did pleased us"), and as object of one of three types of verbs, which include "non-action" verbs ("I know that he came," "I know what he did"), certain transitive verbs with particles ("I complained that he was sick," "I complained about where he went"), and "double-object verbs of 'telling'" ("I told her that he saw us," "I told her who he was"). In all of these cases it should be clear that some verbal relationship is itself being treated as an object that can be talked about. If I were to say, "What lay on the table was clear to us," I would not be referring to the particular concrete object on the table, but to the abstraction of its being there -- this fact is used as information. On the other hand, "What lay on the table was no possession of mine" refers to the concrete object, and I am making a statement about it, not my knowledge of it. (Notice that this makes the statement "I know what he knows" ambiguous; I may know exactly the same things that he does, or I may be aware of the fact that he knows what he does. The latter is the Factive interpretation.)*

Another interesting type of nominal, and one far more useful than the Factive in forming compounds, is introduced by Lees as the "Action" nominal (this nominal has two forms, the "-Ing" form and the "-Nml")

* Most of the Factive examples in this paragraph come directly from [Lees 1963, pp. 59-60].

form). Rather than express the fact that an event has transpired, the Action nominal deals with the way that the event has proceeded. For example, "His drawing of her portrait fascinated me because he did it left-handed" uses the nominal, "drawing", as a reference to the way that a particular event has transpired. Contrast this with the Factive nominal, "That he drew fascinated me because I didn't know he could be persuaded so easily" -- the Factive concerns only the fact of his drawing. The Action nominal can also be used to express habitual actions ("His drawing was always done left-handed")*.

The Action nominal comes in two forms, according to Lees, the "-Ing" form and the "-Nml" form. The -Ing form we have just seen; the -Nml form covers all other nominalized verbs that express this notion of action without "-ing" suffixes. -Nml nominals can be abstract (e.g., repair, conservation, control, retirement, drag) or concrete (e.g., test, report, attempt, ride, recovery, bath, fight).

The two most salient syntactic features of the Action nominal (and remember that these nominals are motivated entirely syntactically -- cf. the Gerundive nominal, below) are 1) the necessary intervention of the preposition "of" between an Action nominal (from a transitive verb) and its direct object ("An understanding of the broad range of nominal expressions . . . first requires an appreciation of the appropriate underlying relationship . . . ,"not "An understanding the broad range . . ."), and 2) the non-preservation of auxiliaries through nominalization (i.e., we cannot say "his having drawn of the portrait" or "his having brought up of the subject").

The final nominal type to be considered here is called by Lees the "Gerundive", and is one that has forms very similar to the -Ing form of the Action nominal. All Gerundives end in "-ing", and can generally be formed freely from any verb (Lees states that certain "non-action" verbs

* These Action examples are also due to Lees [1963, p. 64].

do not have Action nominals). We might rephrase our Factive "That he drew . . ." above to be "His drawing fascinated me because I didn't know he could be persuaded so easily"; Lees calls this the "fact" form of the Gerundive. Notice that the absence of the "of" between nominal and object seems to imply that we are talking about the fact of the situation or event, as in "His driving the car surprised me," as opposed to "His driving of the car gave me motion sickness," which entails the way he did it. The Gerundive nominal does have an "action" form -- this form has no expressed subject, as in "Drawing fascinates me." Notice that the more common "fact" form does preserve auxiliaries in the transformation; it is perfectly reasonable to use the phrases "his having brought up the subject" and "his having drawn the portrait" when talking about the fact that such events occurred. One final note on these "fact" form Gerundives -- they require a subject with a POSS morpheme. We can not say "the driving the car" or "a having brought up the subject" in the manner that we can with Action nominals.

A.2. Fraser's and Chomsky's nominalizations

Some of Lees' original views on nominalization have been disputed, and redisputed, because of problems involved with their being embedded in a transformational framework. Apparently, certain of the nominals (the Gerundive, in particular) are easy to produce transformationally from a source sentence, while others seem not to be transformationally related to propositions in the base component. Chomsky [1970] defends the "lexicalist position" in regard to what he calls "derived" nominals, and Fraser [1970] redefends Lees and the "transformationalist position", at least for his version of the Action nominal. I am not so interested here in these two positions, since the syntactic TG framework is irrelevant to my ultimate purpose, but in the way that these others have characterized the nominals.

Fraser's article concentrates on only the Action nominal, of which he sees two forms. Both of these forms end in "-ing" and look a lot like Lees' -Ing form of the same nominal. And, as he puts it, "The interpretation of these action nominalizations is that of an action, an activity, an act, an event." [1970, p. 84]

The first of Fraser's two forms is characterized by a possessive and the intervening "of" between nominal and direct object: "His figuring out of the solution (took one hour)," "John's riding of the bicycle" The other form has the same "of" construction, but is preceded by a singular article, and is often followed by a "by" phrase containing the subject noun: "The climbing of Mt. Vesuvius by a lone hiker (is an impossible feat)," "(I have never seen) a filming of a motion picture." *

In his detailed investigation of the transformational characteristics of the Action nominal, Fraser mentions in passing two other nominal types which bear resemblance to types that we saw in Section A.1. One is exemplified by the following sentence: "His figuring out the problem (astounded us)." Fraser calls this a "factive nominal", and explains how it should be interpreted as the assertion of a fact, i.e., a statement, not an activity. This is precisely the interpretation of Factives given by Lees; the syntactic type of this nominal is delegated by Lees, however, to the Gerundive ("fact" form) class.

The other nominalization touched upon by Fraser is termed by him the "substantive", and includes the non-Ing type nominals (e.g., refusal, disgust, destruction, etc.). Fraser claims that the interpretation of these nominals is of a completed activity, and thus a fact, as witnessed by, for example, "The U.S.'s destruction of Vietnam (infuriated us)," which talks about the fact of the destruction rather than any

* These examples are from Fraser's article [1970], pp. 84 and 86.

characteristics of the event itself. While Fraser shies away from the substantive because it has "been but barely investigated and few conclusions can be drawn," he illustrates an interesting three-way ambiguity for some transitive verbs -- "John's driving" could represent

- "(1) the fact that John drives;
- (2) some specific activity, for example, John's driving of the car yesterday;
- (3) the general name given to the way in which John operates a motor vehicle"

[1970, p. 85]

The third case shows that some "-ing" forms can be substantives, since "the way in which John drives" is not the source of the Action nominal. Fraser also lets on that some nominals which have the substantive (non-"-ing") form may be interpreted as Action nominals.

Chomsky's focus is on the differences between the Gerundive category and one he calls "derived". While Gerundives can be formed with impunity, and exhibit regular relations between the nominal and its associated proposition, apparently the Derived nominal is not nearly so productive, and exhibits quite varied and idiosyncratic semantic relations to its source proposition. This latter point, says Chomsky, "has been so often remarked that discussion is superfluous." [p. 189] He presents in evidence nominals such as "laughter", "marriage", "construction", "action", "activity", "belief", "doubt", "residence", "qualification", "trial", etc., with their vastly different relationships to source verbals. It is this nominal that Chomsky says cannot be accounted for with a transformational mechanism, but requires, rather, augments to the underlying lexical forms.

Other differences between Gerundive and Derived nominals include the Gerundive's requirement of a subject +POSS, and the fact that Derived nominals have the internal structure of noun phrases (e.g., adjectives can be interjected: "John's unmotivated criticism of the book") while Gerundives do not ("John's unmotivated criticizing the book"). Also, Chomsky mentions only briefly the Action nominal, which he calls "mixed", since some aspects of this form resemble the Derived nominal

(the possessive can be replaced by a determiner -- it exhibits the internal form of a noun phrase), and some do not ("adjective insertion seems quite unnatural"). Chomsky expresses doubts that the lexicalist hypothesis can be extended to cover this form of nominal, which he tosses off as "quite resistant to systematic investigation" and "rather clumsy". On the other hand, as I have mentioned, Fraser claims that the transformationalist hypothesis can be defended for Action nominals.

Table A.1 attempts to integrate these three sources of information about nominals. I do not include the Agentive nominal here, since Lees is the only source of information on it, and it exhibits only a single fixed form.

	Lees (1963)	Fraser (1970)	Chomsky (1970)
the growing of tomatoes	ACTION	ACTION	"Mixed"
His refusing of the offer was done tactlessly.	(-Ing)		
His drawing was always done left-handed.			

His refusing the offer surprised me.	GERUNDIVE ("fact" form)	FACTIVE	GERUNDIVE
His drawing fascinated me because I didn't know he could be persuaded so easily.			

His refusal of the offer ended the negotiations. the growth of tomatoes	ACTION (-Nml)	SUBSTANTIVE	DERIVED

His drawing was huge.	?	?	DERIVED

The apple is for eating the apple.	GERUNDIVE ("action" form)	?	GERUNDIVE?
Running races is good for you.			

His running keeps him in shape.	ACTION (-Ing)	SUBSTANTIVE	"Mixed"?

The destruction of the city was carried out methodically.	ACTION (-Nml)	ACTION (grudgingly)	DERIVED?

Table A.1. Lees, Fraser, and Chomsky on nominals.

Bibliography

- Anderson, John R., and Bower, Gordon H. 1973. Human Associative Memory. Washington, D.C.: V. H. Winston & Sons.
- _____. 1974. "A propositional theory of recognition memory." Memory and Cognition. Vol. 2, No. 3, July, 1974, pp. 406-412.
- Bartlett, Sir Frederick C. 1967. Remembering: A Study in Experimental and Social Psychology. Cambridge: Cambridge University Press (first published 1932).
- Bell, Anthony, and Quillian, M. Ross. 1971. "Capturing concepts in a semantic net." In Associative Information Techniques. E. L. Jacks, ed. New York: American Elsevier Publishing Company.
- Bobrow, Daniel G., and Collins, Allan M. (eds.). 1975. Representation and Understanding: Studies in Cognitive Science. New York: Academic Press.
- Bobrow, Daniel G., and Winograd, Terry. 1977. "An overview of KRL, a knowledge representation language." Cognitive Science, Vol. 1, No. 1, January, 1977, pp. 3-46.
- Brachman, Ronald J. 1973. "Understanding what it takes to understand: System requirements for natural language understanding." Unpublished paper, Applied Mathematics 223, Harvard University.
- _____. 1975. "Structural knowledge in a document information consulting system." TR6-75. Cambridge, MA: Center for Research in Computing Technology, Harvard University.
- _____. 1977. "What's in a concept: Structural foundations for semantic networks." International Journal of Man-Machine Studies. 9, March, 1977, pp. 127-152. Also BBN Report No. 3433. Cambridge, MA: Bolt Beranek and Newman Inc., October, 1976.
- _____. 1978. "On the epistemological status of semantic networks." To appear in Associative Networks -- The Representation and Use of Knowledge in Computers. Nicholas V. Findler, ed. New York: Academic Press. Also BBN Report No. 3807. Cambridge, MA: Bolt Beranek and Newman Inc., April, 1978.
- Brown, David C., and Kwasny, Stan C. 1977. "A natural language graphics system." OSU-CISRC-TR-77-8. Columbus: Computer and Information Science Research Center, The Ohio State University, June, 1977.
- Bruce, Bertram. 1975. "Case systems for natural language." Artificial Intelligence, Vol. 6, No. 4, Winter, 1976, pp. 327-360.

BBN Report No. 3605
Bolt Beranek and Newman Inc.

- Burton, Richard R. 1976. "Semantic grammar: An engineering technique for constructing natural language understanding systems." BBN Report No. 3453 (ICAI Report No. 3). Cambridge, MA: Bolt Beranek and Newman, Inc., December, 1976.
- Carbonell, Jaime R. 1970a. "Mixed-initiative man-computer instructional dialogues." BBN Report No. 1971. Cambridge, MA: Bolt Beranek and Newman, Inc., May 31, 1970.
- _____. 1970b. "AI in CAI: An artificial intelligence approach to computer-aided instruction." IEEE Transactions on Man-Machine Systems, Vol. MMS-11, No. 4, December, 1970.
- Carnap, Rudolf. 1947. Meaning and Necessity. Chicago: The University of Chicago Press.
- Cercone, Nick. 1975. "Representing natural language in extended semantic networks." Technical Report TR75-11. Edmonton, Alberta: Department of Computing Science, The University of Alberta, July, 1975.
- Cercone, Nick, and Schubert, Len. 1975. "Toward a state based conceptual representation." In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, September, 1975, pp. 83-90.
- Charniak, Eugene. 1975. "A brief on case." Working Paper 22. Castagnola, Switzerland: Istituto per gli Studi Semantici e Cognitivi.
- Chomsky, Noam. 1970. "Remarks on nominalization." In Readings in English Transformational Grammar. Roderick A. Jacobs and Peter S. Rosenbaum, eds. Waltham, MA: Ginn and Company, pp. 184-221.
- Codd, E. F. 1970. "A relational model of data for large shared data banks." Communications of the Association for Computing Machinery, Vol. 13, No. 6, June, 1970, pp. 377-387.
- Collins, Allan M., and Loftus, Elizabeth F. 1975. "A spreading-activation theory of semantic processing." Psychological Review, Vol. 82, No. 6, pp. 407-428.
- Collins, Allan M., and Quillian, M. Ross. 1969. "Retrieval time from semantic memory." Journal of Verbal Learning and Verbal Behavior, 8, pp. 240-247.
- _____. 1970a. "Facilitating retrieval from semantic memory: The effect of repeating part of an inference." In Acta Psychologica 33 Attention and Performance III. A. F. Sanders, ed. Amsterdam: North-Holland Publishing Company, pp. 304-314.
- _____. 1970b. "Does category size affect categorization time?" Journal of Verbal Learning and Verbal Behavior, 9, pp. 432-438.

Bibliography

- _____. 1972a. "Experiments on semantic memory and language comprehension." In Cognition in Learning and Memory. L. W. Gregg, ed. New York: J. Wiley and Sons.
- _____. 1972b. "How to make a language user." In Organization of Memory. Endel Tulving and Wayne Donaldson, eds. New York: Academic Press, pp. 309-351.
- Collins, Allan M., and Warnock, Eleanor H. 1974. "Semantic networks." EBN Report No. 2833 (A.I. Report No. 15). Cambridge, MA: Bolt Beranek and Newman, Inc., May, 1974.
- Fahlman, Scott E. 1977. "A system for representing and using real-world knowledge." Ph.D. Thesis Draft. Cambridge, MA: Artificial Intelligence Laboratory, M.I.T., June 15, 1977.
- Fillmore, Charles. 1968. "The case for case." In Universals in Linguistic Theory. Emmon Bach and Robert Harms, eds. New York: Holt, Rinehart and Winston, pp. 1-88.
- Fraser, Bruce. 1970. "Some remarks on the Action Nominalization in English." In Readings in English Transformational Grammar. Roderick A. Jacobs and Peter S. Rosenbaum, eds. Waltham, MA: Ginn and Company, pp. 83-98.
- Gleitman, Lila R., and Gleitman, Henry. 1970. Phrase and Paraphrase. New York: Norton.
- Goldstein, Ira P., and Roberts, R. Bruce. 1977. "Nudge, a knowledge-based scheduling program." To appear in Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Boston, MA.
- Grosz, Barbara J. 1977. "The representation and use of focus in a system for understanding dialogue." In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, August, 1977, pp. 67-76.
- Hawkinson, Lowell. 1975. "The representation of concepts in OWL." In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, pp. 107-114.
- Hayes, Philip J. 1977a. "Some association-based techniques for lexical disambiguation by machine." TR25. Rochester, NY: Computer Science Dept., The University of Rochester, June 1977.
- _____. 1977b. "On semantic nets, frames and associations." In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, August, 1977, pp. 99-107.
- Hayes-Roth, Frederick. 1975. "Collected papers on the learning and recognition of structured patterns." Pittsburgh: Department of Computer Science, Carnegie-Mellon University, January, 1975.

BBN Report No. 3605
Bolt Beranek and Newman Inc.

- Hays, David G. 1973a. "A theory of conceptual organization and processing." Unpublished draft, State University of New York at Buffalo.
- _____. 1973b. "Types of processes on cognitive networks."
Paper presented at the 1973 International Conference on Computational Linguistics, Pisa, Italy, August 27-September 1, 1973.
- Heidorn, George E. 1972. "Natural language inputs to a simulation programming system." NPS-55HD72101a. Monterey, CA: Naval Postgraduate School, October, 1972.
- _____. 1974. "English as a very high level language for simulation programming." ACM SIGPLAN Notices, Volume 9, Number 4, April, 1974, pp 91-100.
- Hendrix, Gary G. 1975a. "Partitioned networks for the mathematical modeling of natural language semantics." Technical Report NL-28. Austin, TX.: Dept. of Computer Sciences, The University of Texas at Austin, December, 1975.
- _____. 1975b. "Expanding the utility of semantic networks through partitioning." In Proceedings of the Fourth International Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, pp. 115-121.
- _____. 1976. "The representation of semantic knowledge." In Speech Understanding Research: Final Technical Report. Donald E. Walker, ed. Menlo Park, CA: Stanford Research Institute, October 1976.
- _____. 1978. "Encoding knowledge in partitioned networks." To appear in Associative Networks -- The Representation and Use of Knowledge in Computers. Nicholas V. Findler, ed. New York: Academic Press.
- Hendrix, Gary G., Thompson, Craig W., and Slocum, Jonathan. 1973. "Language processing via canonical verbs and semantic models." In Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, CA, pp. 262-269.
- Irwin, J., and Srinivasan, C. V. 1975. "Description of CASNET in MDS." Report CBM-TR-49. New Brunswick, NJ: Department of Computer Science, Rutgers University, August, 1975.
- Landsbergen, S. P. J. 1976. "Syntax and formal semantics of English in PHLQA1." COLING '76 Preprints, Ottawa, Ontario.
- Lees, Robert B. 1963. The Grammar of English Nominalizations. The Hague, The Netherlands: Mouton and Co., Publishers.
- _____. 1970. "On very deep grammatical structure." In Readings in English Transformational Grammar. Roderick A. Jacobs and Peter S. Rosenbaum, eds. Waltham, MA: Ginn and Company, pp. 134-142.

Bibliography

- Levesque, Hector J. 1977. "A procedural approach to semantic networks." Technical Report No. 105. Toronto: Dept. of Computer Science, University of Toronto, April 1977.
- Levesque, Hector J., and Mylopoulos, John. 1978. "A procedural semantics for semantic networks." To appear in Associative Networks -- The Representation and Use of Knowledge in Computers. Nicholas V. Findler, ed. New York: Academic Press.
- Lindsay, Robert. 1973. "In defense of ad hoc systems." In Computer Models of Thought and Language. Roger C. Schank and Kenneth Mark Colby, eds. San Francisco: W. H. Freeman and Co., pp. 372-395.
- McDermott, Drew. 1974. "Assimilation of new information by a natural language understanding system." AI TR-291. Cambridge, MA: Artificial Intelligence Laboratory, M.I.T., February, 1974.
- _____. 1976. "Artificial intelligence meets natural stupidity." SIGART Newsletter, No. 57, April, 1976, pp. 4-9.
- Martin, William A. 1975. "Conceptual grammar." Internal Memo 20, Automatic Programming Group, Project MAC, M.I.T., September 15, 1975.
- _____. 1977. "OWL." In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, August, 1977, pp. 985-987.
- Minsky, Marvin (ed.). 1968. Semantic Information Processing. Cambridge, MA: The M.I.T. Press.
- _____. 1975. "A framework for representing knowledge." In The Psychology of Computer Vision. Patrick H. Winston, ed. New York: McGraw-Hill Book Company, pp. 211-277.
- Moore, J., and Newell, A. 1973. "How can Merlin understand?" In Knowledge and Cognition. L. Gregg, ed. Potomac, MD: Lawrence Erlbaum Associates.
- Myer, Theodore H., Mooers, Charlotte D., and Stevens, Albert L. 1977. Hermes Users' Guide (Revised Edition). Cambridge, MA: Bolt Beranek and Newman, Inc., February, 1977.
- Neisser, Ulric. 1967. Cognitive Psychology. New York: Appleton-Century-Crofts.
- Norman, Donald A. 1972. "Memory, knowledge, and the answering of questions." CHIP Technical Report 25. La Jolla, CA: Center for Human Information Processing, University of California, San Diego, May, 1972.
- _____. 1973. "Learning and remembering: A tutorial preview." In Attention and Performance IV. S. Kornblum, ed. New York: Academic Press.

BBN Report No. 3605
Bolt Beranek and Newman Inc.

- Norman, Donald A., Rumelhart, David E., and the LNR Research Group.
1975. Explorations in Cognition. San Francisco: W. H. Freeman and Co.
- Quillian, M. Ross. 1966. "Semantic memory." Report AFCRL-66-189.
Cambridge, MA: Bolt Beranek and Newman, Inc., October, 1966.
- _____. 1967. "Word concepts: A theory and simulation of some basic semantic capabilities." Behavioral Science, Vol. 12, No. 5, September, 1967, pp. 410-430.
- _____. 1968. "Semantic memory." In Semantic Information Processing. Marvin Minsky, ed. Cambridge, MA: The M.I.T. Press, pp. 227-270.
- _____. 1969. "The Teachable Language Comprehender: A simulation program and theory of language." Communications of the Association for Computing Machinery, Vol. 12, No. 8, August, 1969, pp. 459-476.
- Quine, Willard Van Orman. 1953. From a Logical Point of View. Cambridge, MA: Harvard University Press.
- _____. 1960. Word and Object. Cambridge, MA: The M.I.T. Press.
- Raphael, Bertram. 1968. "SIR: Semantic Information Retrieval." In Semantic Information Processing. Marvin Minsky, ed. Cambridge, MA: The M.I.T. Press, pp. 33-145.
- Rhyme, James. 1975. "A lexical process model of nominal compounds in English." American Journal of Computational Linguistics, Microfiche 33:33-44.
- Rieger, Chuck. 1975. "The Commonsense Algorithm as a basis for computer models of human memory, inference, belief and contextual language comprehension." In Proceedings of the Workshop on Theoretical Issues in Natural Language Processing, Bonnie L. Nash-Webber and Roger Schank, eds. Cambridge, MA, June, 1975, pp. 180-195.
- _____. 1976. "An organization of knowledge for problem solving and language comprehension." Artificial Intelligence, Volume 7, Number 2, Summer, 1976, pp. 89-127.
- _____. 1977. "Spontaneous computation in cognitive models." Cognitive Science, Volume 1, Number 3, July, 1977, pp. 315-354.
- Rieger, Chuck, and Grinberg, Milt. 1977. "The declarative representation and procedural simulation of causality in physical mechanisms." In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, August, 1977, pp. 250-256.
- Roberts, R. Bruce, and Goldstein, Ira P. 1977. FRL Users' Manual. A.I. Memo No. 408. Cambridge, MA: Artificial Intelligence Laboratory, M.I.T., April, 1977.

Bibliography

- Rumelhart, David E., Lindsay, Peter H., and Norman, Donald A. 1972. "A process model for long-term memory." In Organization of Memory. Endel Tulving and Wayne Donaldson, eds. New York: Academic Press, pp. 197-246.
- Rumelhart, David E., and Norman, Donald A. 1973. "Active semantic networks as a model of human memory." In Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, CA, pp. 450-457.
- Rustin, Randall (ed.). 1973. Natural Language Processing. New York: Algorithmics Press.
- Schank, Roger C. 1972. "Conceptual dependency: A theory of natural language understanding." Cognitive Psychology, 3, pp. 552-631.
- _____. 1973a. "The conceptual analysis of natural language." In Natural Language Processing. Randall Rustin, ed. New York: Algorithmics Press, pp. 291-309.
- _____. 1973b. "Identification of conceptualizations underlying natural language." In Computer Models of Thought and Language. Roger C. Schank and Kenneth Mark Colby, eds. San Francisco: W. H. Freeman and Co., pp. 187-247.
- _____. 1974. "Is there a semantic memory?" Technical Report 3. Castagnola, Switzerland: Istituto per gli Studi Semantici e Cognitivi, March, 1974.
- _____. 1975. "The structure of episodes in memory." In Representation and Understanding: Studies in Cognitive Science. Daniel G. Bobrow and Allan M. Collins, eds. New York: Academic Press, pp. 237-272.
- Schank, Roger C., and Colby, Kenneth Mark (eds.). 1973. Computer Models of Thought and Language. San Francisco: W. H. Freeman and Co.
- Schank, Roger C., Goldman, Neil, Rieger, Charles J. III, and Riesbeck, Chris. 1973. "MARGIE: Memory, Analysis, Response Generation, and Inference on English." In Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, CA, pp. 255-261.
- Schank, Roger C., and Rieger, Charles J. III. 1974. "Inference and the computer understanding of natural language." Artificial Intelligence, Volume 5, Number 4, Winter, 1974, pp. 373-412.
- Schubert, L. K. 1976. "Extending the expressive power of semantic networks." Artificial Intelligence, Vol. 7, No. 2, Summer, 1976, pp. 163-198.
- Scrugg, Greg W. 1975. "Frames, planes, and nets: A synthesis." Working paper 19. Castagnola, Switzerland: Istituto per gli Studi Semantici e Cognitivi.

BBN Report No. 3605
Bolt Beranek and Newman Inc.

- Shapiro, Stuart C. 1971a. "The MIND system: A data structure for semantic information processing." Technical Report R-837-PR. The Rand Corporation, August, 1971.
- _____. 1971b. "A net structure for semantic information storage, deduction, and retrieval." In Proceedings of the Second International Joint Conference on Artificial Intelligence pp. 512-523.
- _____. 1975. "An introduction to SNePS." Bloomington: Computer Science Department, Indiana University, March, 1975.
- _____. 1977. "Representing and locating deduction rules in a semantic network." SIGART Newsletter, No. 63, June, 1977, pp. 14-18.
- _____. 1978. "The SNePS semantic network processing system." To appear in Associative Networks -- The Representation and Use of Knowledge in Computers. Nicholas V. Findler, ed. New York: Academic Press.
- Simmons, Robert F. 1973. "Semantic networks: Their computation and use for understanding english sentences." In Computer Models of Thought and Language. Roger C. Schank and Kenneth Mark Colby, eds. San Francisco: W. H. Freeman and Co., pp. 63-113.
- Simmons, Robert F., and Bruce, Bertram C. 1971. "Some relations between predicate calculus and semantic net representations of discourse." In Proceedings of the Second International Joint Conference on Artificial Intelligence, pp. 524-529.
- Simmons, Robert F., Burger, John F., and Schwarcz, Robert M. 1968. "A computational model of verbal understanding." AFIPS Conference Proceedings, Vol. 33 (1968 Fall Joint Computer Conference), pp. 441-456.
- Simmons, Robert F., and Chester, Daniel. "Inferences in quantified semantic networks." In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, pp. 267-273.
- Simmons, Robert F., and Slocum, J. 1972. "Generating English discourse from semantic networks." Communications of the Association for Computing Machinery, Vol. 15, No. 10, pp. 891-905.
- Smith, Brian C. 1977. Lecture on "The representation semantics of KRL." Bolt Beranek and Newman, Inc., February 11, 1977.
- _____. 1978. "Levels, layers, and planes: The framework of a system of knowledge representation semantics." Master's thesis. Cambridge, MA: Artificial Intelligence Laboratory, M.I.T., January, 1978.
- Srinivasan, Chitoor V. 1976. "The architecture of Coherent Information System: A general problem solving system." IEEE Transactions on Computers, Vol. C-25, No. 4, April, 1976, pp. 390-402.

Bibliography

- Szolovits, Peter, Hawkinson, Lowell B., and Martin, William A. 1977. "An overview of OWL, a language for knowledge representation." MIT/LCS/TM-86. Cambridge, MA: Laboratory for Computer Science, M.I.T., June, 1977.
- Tulving, Endel, and Donaldson, Wayne (eds.). 1972. Organization of Memory. New York: Academic Press.
- Warnock, Eleanor H., and Collins, Allan M. 1973. "Research on Semantic Nets." Cambridge, MA: Bolt Beranek and Newman, Inc., May 8, 1973.
- Wilks, Yorick. 1974. "Natural language understanding systems within the AI paradigm." Memo AIM-237. Stanford, CA: Stanford Artificial Intelligence Laboratory, Stanford University, December, 1974.
- Winograd, Terry. 1975. "Frame representations and the declarative/procedural controversy." In Representation and Understanding: Studies in Cognitive Science. Daniel G. Bobrow and Allan M. Collins, eds. New York: Academic Press, pp. 185-210.
- Winston, Patrick H. 1970. "Learning structural descriptions from examples." Project MAC TR-76. Cambridge, MA: M.I.T.
- _____. (ed.). 1975. The Psychology of Computer Vision. New York: McGraw-Hill Book Company.
- Woods, William A. 1968. "Procedural semantics for a question-answering machine." AFIPS Conference Proceedings, Vol. 33 (1968 Fall Joint Computer Conference), pp. 457-471.
- _____. 1975a. "What's in a link: Foundations for semantic networks." In Representation and Understanding: Studies in Cognitive Science, Daniel G. Bobrow and Allan M. Collins, eds. New York: Academic Press, pp. 35-82.
- _____. 1975b. "MNEMO: A memory machine for associative factual retrieval." Preliminary manuscript, March, 1975.

Official Distribution List
Contract N00014-77-C-0371

	<u>Copies</u>
Scientific Officer Program Director, Information Systems Office of Naval Research 800 North Quincy Street Arlington, VA 22217 Attn: Gordon Goldstein, Code 437	1
Administration Contracting Officer Defense Contracting Administration Services 666 Summer Street Boston, MA 02210	1
Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20375	6
Office of Naval Research Department of the Navy Code 1021P Arlington, VA 22217	6
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	1